GUIDE PRATIQUE

METHODES DE CALCUL NUMERIQUE

PROGRAMMES EN BASIC

CLAUDE NOWAKOWSKI

TOME 1

EDITIONS DU P.S.I.

MÉTHODES DE CALCUL NUMÉRIQUE

PROGRAMMES EN BASIC

Collection Langages

Le langage ADA - Daniel-Jean David
Programmer en APL - Daniel-Jean David
Programmer en Assembleur - Alain Pinaud
Programmer en BASIC - Michel Plouin
Le BASIC et ses fichiers - Tomes 1 et 2 - Jacques Boisgontier
Programmer en Fortran - Daniel-Jean David
Programmer en L.S.E. - Stéphane Berche et Yves Noyelle
Programmer en PASCAL - Daniel-Jean David et Jean-Luc Deschamps
Comment programmer - Jean-Claude Barbance
Comprendre les microprocesseurs - Roland Dubois

Collection Matériels

La découverte de l'Applesoft - Tome 1 - Frédéric Lévy et Dominique Schraen
La découverte de l'Applesoft - Tome 2 - Frédéric Lévy
La pratique de l'Apple II - Volumes I et II - Nicole Bréaud-Pouliquen
La pratique de l'Apple II - Volume III - Nicole Bréaud-Pouliquen et Daniel-Jean David
La pratique du LX-500 - Alain Séméteys et Francis Vasse
La pratique du MZ 80-K - Jean-Pierre Lhoir
La découverte du PC-1211 - Jean-Pierre Richard
La découverte du PET/CBM - Daniel-Jean David
La pratique du PET/CBM - Volumes I et II - Daniel-Jean David
La pratique du TRS-80 - Volumes I, II et III - Pierre Giraud et Alain Pinaud
La découverte de le TI-57 - Xavier de La Tullaye
La découverte du VIC - Daniel-Jean David

Collection Guides Pratiques

La réalisation des programmes - Michel Benelfoul
Méthodes de calcul numérique - Claude Nowakowski

Collection Programmes

Jeux, trucs et comptes pour PET/CBM - Michel Benelfoul
Mathématiques et statistiques - Hervé Haut
Modèles pratiques de décision - Jean-Pierre Blanger
Récréations pour TI-57 - Jacques Deconchat
Variations pour PC-1211 - Jean-François Sehan

Collection Mémentos

Clefs pour le B.E.T./C.B.M. - Daniel-Jean David

Traductions

Les graphiques sur TRS-80 - Don Inman traduit par Alain Pinaud
Le petit livre du ZX-81 - Trevor Toms traduit par Jean-Pierre Richard

---

INITIATION

VISA POUR L'INFORMATIQUE - Jean-Michel Jégo
MON ORDINATEUR - Jean-Claude Barbance

La loi du 11 mars 1957 n'autorisant, aux termes des alinéas 2 et 3 de l'article 41, d'une part, que les «copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective» et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, «toute représentation ou reproduction intégrale, ou partielle, faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause, est illicite» (alinéa 1er de l'article 40).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du Code Pénal.

ISBN : 2-86595-029-8

© Editions du P.S.I., 41-51 rue Jacquard, B.P. 86, 77400 Lagny-sur-Marne (France)
1981

Imprimé en France

GUIDE PRATIQUE

MÉTHODES DE CALCUL NUMÉRIQUE

PROGRAMMES EN BASIC

TOME 1

CLAUDE NOWAKOWSKI

EDITIONS DU P.S.I.
1982

S O M M A I R E

		Pages
INTRODUCTION		7
CHAPITRE I	**EQUATIONS NON LINEAIRES**	9
	Généralités	9
	Méthode de la dichotomie	11
	Méthode de la sécante	15
	Méthode de Newton	18
	Compléments théoriques	20
	Systèmes d'équations non linéaires	22
CHAPITRE II	**POLYNOMES**	29
	Introduction	29
	Opérations polynomiales	30
	Racines d'un trinôme	32
	Méthode itérative de Birge-Vieta	33
	Méthode itérative de Bairstow	37
	Méthode de Bernouilli	45
CHAPITRE III	**CALCUL MATRICIEL : 1 - SYSTEMES D'EQUATIONS LINEAIRES**	49
	Introduction	49
	Résolution des systèmes par élimination	50
	Méthode de Crout-Décomposition	56
	Inversion de matrices	60
CHAPITRE IV	**CALCUL MATRICIEL : 2 - VALEURS ET VECTEURS PROPRES METHODES ITERATIVES POUR SYSTEMES LINEAIRES**	71
	Valeurs et vecteurs propres	71
	Méthode de la puissance itérée	72
	Déflation de la matrice	76
	Algorithme de Rutishauser	81
	Méthodes itératives pour les systèmes linéaires	84
CHAPITRE V	**INTERPOLATION**	89
	Polynôme de Lagrange	89
	Formule d'interpolation de Newton	94
	Algorithme d'Aitken	102

		Pages
CHAPITRE VI	**INTEGRATION**	107
	Rappel	107
	Méthode des coefficients indéterminés	109
	Formules de Newton Côtes et méthodes composites	114
	Méthode de Gauss	119
	Méthode de Romberg	128
CHAPITRE VII	**EQUATIONS DIFFERENTIELLES**	131
	Problème de Cauchy	131
	Méthode d'Euler	133
	Etude sommaire des méthodes à un pas	137
	Méthodes de Runge Kutta	139
	Systèmes différentiels	143
CHAPITRE VIII	**APPLICATIONS**	147
	Réseaux électriques en courant continu	147
	Charge d'un condensateur	149
BIBLIOGRAPHIE SOMMAIRE		155

INTRODUCTION

Les premiers ordinateurs - *machines à calculer automatiques à circuits électroniques et à programme enregistré* - ont permis de traiter des problèmes scientifiques dont les solutions n'étaient pas accessibles.

Depuis, le matériel et le logiciel se sont considérablement développés : les ordinateurs actuels ont des performances qu'il est difficile de bien percevoir. Ainsi, les calculs se font de plus en plus rapidement et les capacités des mémoires deviennent énormes avec des temps d'accès décroissants.

Parallèlement, l'analyse numérique a débouché vers des méthodes nouvelles qui élargissent encore le domaine du calcul scientifique et technique, par suite, les applications sont très variées : conception assistée par ordinateur, simulation, conduite d'installations industrielles, recherche opérationnelle ...

Les langages de programmation, à la fois évolués et simples à utiliser, facilitent sérieusement l'écriture et surtout, la mise au point des programmes.

Mais, aujourd'hui, ce qui est vraiment spectaculaire par les performances, et qui va inévitablement changer l'enseignement et ultérieurement les méthodes de travail, c'est le *Petit Système Individuel (P.S.I.)* qui met à la

portée de beaucoup d'amateurs les moyens matériels de résoudre facilement et rapidement une infinité de problèmes particulièrement complexes. En outre, ces utilisateurs bénéficient de l'expérience d'une quarantaine d'années et ils ont ainsi entre les mains un outil extraordinaire dont souvent les possibilités ne sont pas bien connues.

L'étude des méthodes numériques est très utile pour les ingénieurs et techniciens mais aussi pour tous ceux qui souhaitent voir le lien entre les mathématiques et le micro-ordinateur.

Cet ouvrage se limite à la présentation des méthodes classiques de calcul numérique. Il ne s'agit ni d'un traité d'analyse numérique destiné aux scientifiques, ni un recueil de programmes ou recettes. C'est une initiation au calcul numérique où, pour un même problème, les différentes méthodes utilisables sont présentées en abordant d'abord les plus simples.

Chaque algorithme est illustré par un organigramme résumant la succession des opérations, un programme BASIC (testé sur TRS-80, - Modèle 1) et un exemple numérique (jeu d'essai).

Il peut être assez surprenant de constater que, pour un problème donné, plusieurs méthodes sont proposées : chaque méthode présente évidemment des avantages et des inconvénients, et le choix d'un algorithme demande des essais, une adaptation, voire une expérimentation, pour bien connaître la précision et les temps de calcul.

L'études des sujets retenus (équations non linéaires, polynômes, systèmes linéaires, valeurs propres, intégration et équations différentielles) est une initiation qui peut être approfondie en consultant des ouvrages traitant les mêmes sujets ou bien une première étape pour pouvoir aborder d'autres problèmes bien plus compliqués (problèmes aux limites, équations aux dérivées partielles, analyse harmonique ...).

Enfin, deux exemples, concernant des circuits électriques élémentaires montrent les possibilités de quelques méthodes numériques.

EQUATIONS NON LINEAIRES

GENERALITES

Très souvent, les équations à résoudre ne sont pas linéaires ; les exemples ci-dessous montrent la variété de ce type de problème :

- calculer $\sqrt[n]{x}$
- trouver les racines du polynôme de degré 6 : x^6+5x^4-x+7
- trouver les valeurs de x qui annulent l'expression $x-e \sin x$

Dans tous les cas, le problème se ramène à rechercher la ou les racines d'une équation de la forme :

$$f(x)=0$$

Le cas particulier des polynômes est traité au Chapitre II.

Les méthodes qui conduisent aux valeurs exactes (méthodes analytiques), sont rares et limitées à des cas très simples comme le calcul des racines d'un trinôme du second degré.

Généralement, la ou les racines ne peuvent être calculées que par approximations successives, les calculs sont alors arrêtés lorsque la précision désirée est atteinte. *Ces méthodes numériques sont dites itératives* et l'itération correspond à l'application répétée d'une ou plusieurs formules récurrentes permettant de calculer une meilleure approximation à partir de la ou des valeurs précédemment calculées.

Ces formules sont de la forme :

$$x_{i+1} = F(x_i, x_{i-1}\ldots)$$

où x_{i+1} désigne la nouvelle valeur.

La suite des nombres x_1, $x_2 \ldots x_i$, $x_{i+1}, \ldots$ converge vers la solution α de l'équation $f(x)=0$: x_i se rapproche de α quand i augmente ($x_i \rightarrow \alpha$ quand $i \rightarrow \infty$).

Comme la solution exacte n'est pas connue, le résultat sera considéré comme satisfaisant lorsque la condition :

$$|x_{i+1}-x_i| < \varepsilon$$

sera remplie (x_i est la valeur obtenue au bout de i itérations et i+1 la valeur obtenue à l'itération suivante. ε est l'erreur admise).

Parfois, par suite des erreurs de troncature (✱), il n'est pas possible d'atteindre la précision souhaitée. Dans ce cas, il faut effectuer les calculs avec davantage de chiffres significatifs ou employer d'autres méthodes.

Cela explique pourquoi il n'y a pas une méthode universelle mais de nombreuses.

Les méthodes simples donnent de bons résultats pour presque tous les problèmes, mais elles convergent lentement : il faut faire de nombreux calculs pour obtenir la précision requise (les programmes compilés sont ici bien plus efficaces que les programmes interprétés).

(✱) le nombre de chiffres significatifs est limité !

METHODE DE LA DICHOTOMIE

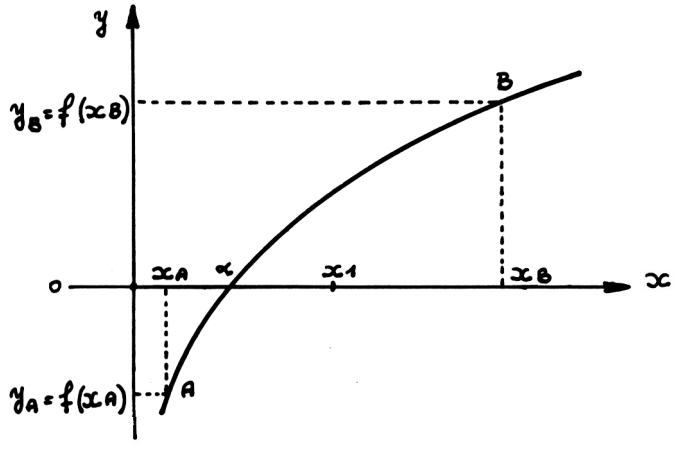
Le nom est compliqué pour une *méthode élémentaire qui correspond à un "tâtonnement rationalisé".*

Il faut d'abord choisir deux valeurs x_A et x_B encadrant la racine α (l'intervalle peut être assez grand), et en conséquence :

$$f(x_A).f(x_B) < 0$$

Les valeurs prises par la fonction $f(x_A)$ pour $x=x_A$ et $f(x_B)$ pour $x=x_B$ doivent être de signes contraires.

La représentation graphique de la fonction montre que les points A et B de la courbe (respectivement aux abscisses x_A et x_B) doivent être choisis de part et d'autre de l'axe 0_X.



Au départ, l'erreur est la plus grande valeur des écarts $\varepsilon_1=|x_A-\alpha|$ et $\varepsilon_2=|x_B-\alpha|$ et au plus $\varepsilon_{max}=|x_B-x_A|$. L'intervalle $[x_A,x_B]$ est alors divisé en 2 :

$$\frac{x_A+x_B}{2} = x_1$$

le point obtenu est plus près de la racine (l'erreur a diminué), c'est la première itération.

Pour le calcul suivant, il faut retenir x_A ou x_B pour continuer à encadrer la racine, en prenant x_1 comme autre borne du nouvel intervalle. Donc, il faut choisir A ou B tel que :

$$f(x_k).f(x_1) < 0$$

k étant A ou B.

Le calcul est le même pour x_2, x_3, x_4... : l'intervalle est, à chaque itération, divisé en 2 (dichotomie).

En résumé :

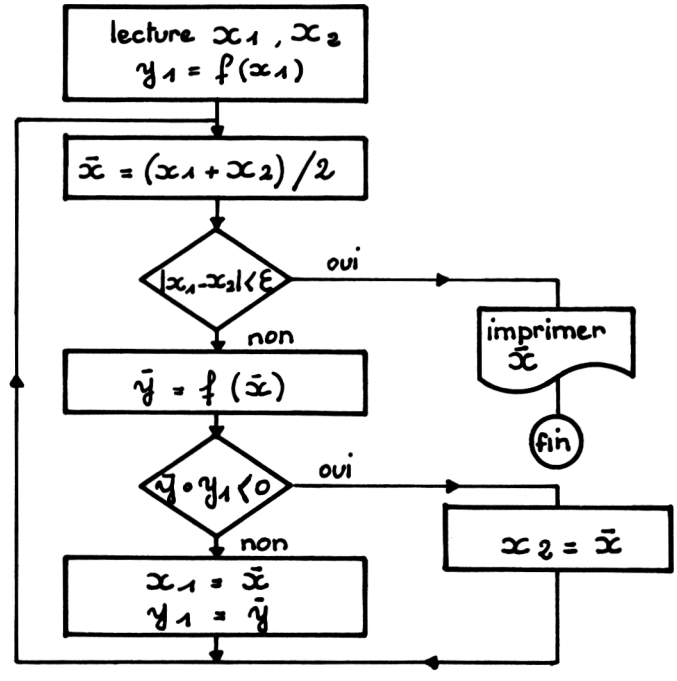
- choisir deux valeurs x_1 et x_2 telles que :
$$f(x_1).f(x_2) < 0$$
- calculer une nouvelle approximation :
$$\bar{x} = \frac{x_1+x_2}{2}$$
- si $|x_1-x_2| < \varepsilon$, alors la précision est atteinte et imprimer le résultat $\bar{x}$, sinon :
- si $f(x_1).f(\bar{x}) < 0$, alors $x_2=\bar{x}$, sinon : $x_1=\bar{x}$
- retourner au calcul de $\bar{x}$ (b)

La méthode numérique (algorithme) peut être représentée par un organigramme qui a l'avantage de montrer les différents traitements suivant les cas rencontrés en cours de calcul.

Remarques :

1 - L'organigramme ci-après présente un défaut important : si la précision ne peut être atteinte, les itérations continuent indéfiniment. Il faut donc modifier le programme pour éviter un "bouclage".

2 - Si les points de départ x_1 et x_2 sont mal choisis : $f(x_1).f(x_2) > 0$ ou bien $|f(x_1).f(x_2)| < \varepsilon$, la méthode ne converge pas, c'est pourquoi il faut d'abord faire ces tests avant de commencer les calculs.

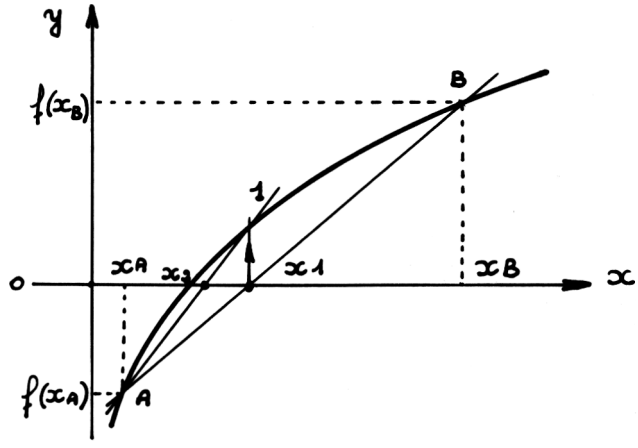



```
5 REM EQUATION NON LINEAIRE ** DICHOTOMIE
10 INPUT "X1,X2";X1,X2
15 LPRINT "VALEURS DE DEPART  ";X1,X2
20 INPUT "ERREUR MAXI";E
25 LPRINT "ERREUR MAXI  ";E
30 X=X1:GOSUB 1000:Y1=Y
40 X=X2:GOSUB 1000:Y2=Y
50 IFY1*Y2)0THEN PRINT "MAUVAIS CHOIX X1,X2":END
55 IF ABS(Y1*Y2)(1E-5 THEN 140
60 FOR K=1 TO 50
70 XB=(X1+X2)/2
80 IF ABS(X1-XB)(E THEN 125
90 X=XB:GOSUB 1000:YB=Y
100 IF Y1*YB(0 THEN X2=XB ELSE X1=XB:Y1=YB
105 PRINT Y
110 NEXT K
120 PRINT "NON CONVERGENCE  X1=";X1,"X2=";X2:END
125 LPRINT"RACINE DE Y=X-2.71828*SIN(X)
130 LPRINT:LPRINT "RACINE";XB
135 LPRINT "NB ITERATIONS ";K:LPRINT "Y= ";Y:END
140 PRINT "X1 OU X2 RACINE";Y1*Y2:END
1000 Y=X-2.71828*SIN(X):RETURN
```

```
VALEURS DE DEPART    1              10
ERREUR MAXI    .1
RACINE DE Y=X-2.71828*SIN(X)
RACINE 2.19531
NB ITERATIONS  7
Y=  .177542

VALEURS DE DEPART    1              10
ERREUR MAXI    .01
RACINE DE Y=X-2.71828*SIN(X)
RACINE 2.2041
NB ITERATIONS  10
Y=  .035974

VALEURS DE DEPART    1              10
ERREUR MAXI    1E-03
RACINE DE Y=X-2.71828*SIN(X)
RACINE 2.19916
NB ITERATIONS  14
Y= -1.33538E-03

VALEURS DE DEPART    1              10
ERREUR MAXI    1E-04
RACINE DE Y=X-2.71828*SIN(X)
RACINE 2.19909
NB ITERATIONS  17
Y= -2.6536E-04


VALEURS DE DEPART    1              10
ERREUR MAXI    1E-05
RACINE DE Y=X-2.71828*SIN(X)
RACINE 2.19911
NB ITERATIONS  20
Y= -4.22001E-05
```

METHODE DE LA SECANTE

Pour cette méthode qui s'appelle aussi méthode de Lagrange, la fonction f(x) est remplacée, dans un intervalle assez petit, par une sécante qui passe par les points A et B et telle que $f(x_A).f(x_B) < 0$



La sécante coupe alors l'axe 0_X en un point x_1 qui définit ainsi le point 1 sur la courbe, en $y_1=f(x_1)$. Puis, une nouvelle sécante passant par A et 1, coupe l'axe 0_X en x_2 et ainsi de suite. C'est le principe de la méthode de la sécante.

Cette méthode ne converge que dans certaines conditions exposées plus loin.

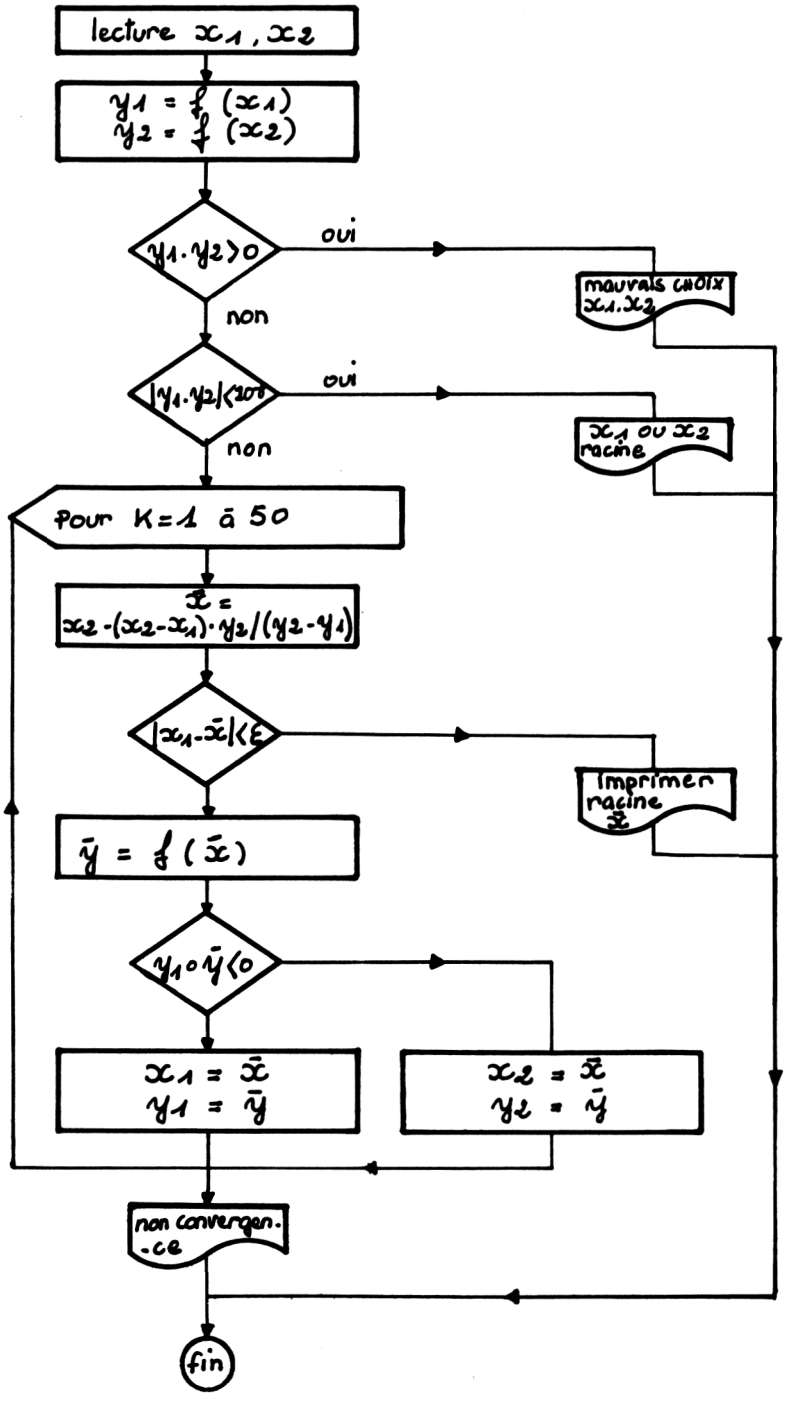
Pour obtenir la formule récurrente, on considère l'équation de la droite passant par les points i et A :

$$\frac{f(x_i)-y}{x_i-x} = \frac{f(x_i)-f(x_A)}{x_i-x_A}$$

L'intersection avec l'axe 0_X a lieu pour y=0 ($x=x_{i+1}$), d'où :

$$x_{i+1} = x_i - \frac{(x_i-x_A)\ f(x_i)}{f(x_i)-f(x_A)}$$

L'organigramme ci-après traduit cet algorithme.


lecture x_1, x_2
$y_1 = f(x_1)$
$y_2 = f(x_2)$
$y_1 . y_2 > 0$
oui
mauvais choix x_1, x_2
non
$|y_1 . y_2| < 10^{-6}$
oui
x_1 ou x_2 racine
non
Pour K=1 à 50
$\bar{x} = x_2 - (x_2 - x_1) \cdot y_2 / (y_2 - y_1)$
$|x_1 - \bar{x}| < \varepsilon$
Imprimer racine $\bar{x}$
$\bar{y} = f(\bar{x})$
$y_1 \circ \bar{y} < 0$
$x_1 = \bar{x}$
$y_1 = \bar{y}$
$x_2 = \bar{x}$
$y_2 = \bar{y}$
non convergen. ce
fin

```
5 REM EQUATION NON LINEAIRE ** LAGRANGE
10 INPUT "X1,X2";X1,X2
15 LPRINT "VALEURS DE DEPART ";X1,X2
20 INPUT "ERREUR MAXI";E
25 LPRINT "ERREUR MAXI ";E
30 X=X1:GOSUB 1000:Y1=Y
40 X=X2:GOSUB 1000:Y2=Y
50 IFY1*Y2)0THEN PRINT "MAUVAIS CHOIX X1,X2":END
55 IF ABS(Y1*Y2)(1E-5 THEN 140
60 FOR K=1 TO 50
70 XB=X2-(X2-X1)*Y2/(Y2-Y1)
80 IF ABS(X1-XB)(E THEN 130
90 X=XB:GOSUB 1000:YB=Y
100 IF Y1*YB(0 THEN X2=XB:Y2=YB ELSE X1=XB:Y1=YB
110 NEXT K
120 PRINT "NON CONVERGENCE  X1=";X1,"X2=";X2:END
125 LPRINT "RACINE DE Y=X-2.71828*SIN(X)
130 LPRINT:LPRINT "RACINE";XB
135 LPRINT "NB ITERATIONS ";K:LPRINT "Y= ";Y:END
140 PRINT"X1 OU X2 RACINE";Y1*Y2:END
1000 Y=X-2.71828*SIN(X):RETURN

VALEURS DE DEPART  1              10
ERREUR MAXI  .1
RACINE 2.19797
NB ITERATIONS  4
Y= -.0516245

VALEURS DE DEPART  1              10
ERREUR MAXI  .01
RACINE 2.19906
NB ITERATIONS  5
Y= -2.98262E-03

VALEURS DE DEPART  1              10
ERREUR MAXI  1E-03
RACINE 2.19912
NB ITERATIONS  6
Y= -1.69277E-04

VALEURS DE DEPART  1              10
ERREUR MAXI  1E-04
RACINE 2.19912
NB ITERATIONS  6
Y= -1.69277E-04

VALEURS DE DEPART  1              10
ERREUR MAXI  1E-05
RACINE 2.19912
NB ITERATIONS  7
Y= -9.53674E-06
```

METHODE DE NEWTON

Cette méthode est plus "fine" et nécessite des calculs un peu plus compliqués, il faut calculer à chaque étape la valeur prise par la fonction au point x_i et la valeur de la dérivée pour cette abscisse x_i.

La racine est approchée par des tangentes à la courbe : à partir du point $(x0,y0)$, on trace la tangente à la courbe, elle coupe l'axe $0x$ en $x1$ qui donne ainsi une meilleure approximation et on recommence ainsi de suite jusqu'à ce que la précision désirée soit atteinte.

L'équation de la droite, tangente à la courbe $y=f(x)$, au point x_i est donnée par :

$$\frac{f(x_i)-y}{x_i-x} = f'(x_i)$$

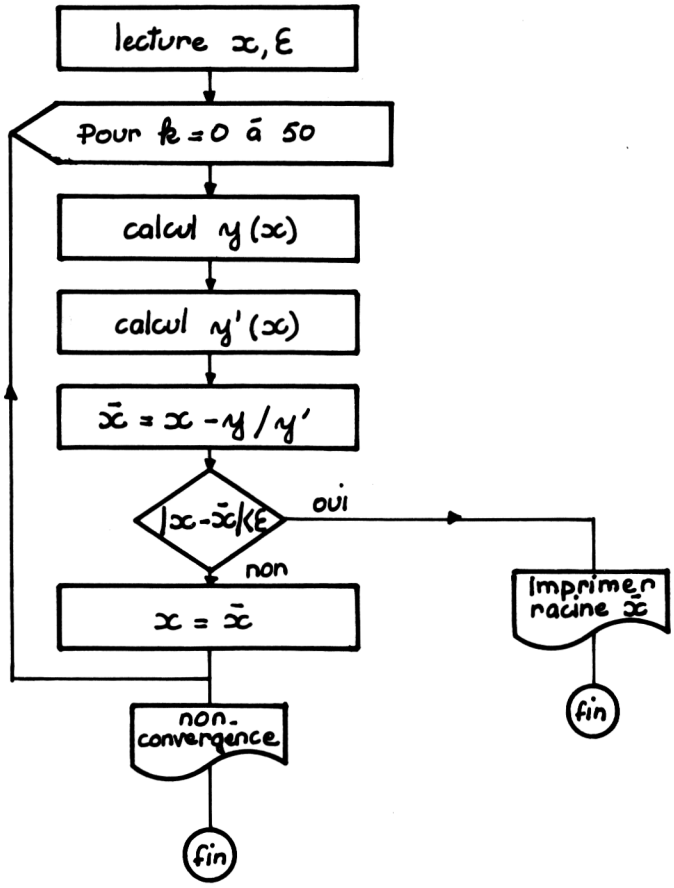
L'intersection avec l'axe $0x$ a lieu quand $y=0$, ce qui donne le point x_{i+1} :

$$\frac{f(x_i)}{x_i-x_{i+1}} = f'(x_i)$$

d'où :

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Evidemment, cette méthode ne peut être appliquée que si $f'(x_i) \neq 0$: il faut que la racine recherchée soit simple.




```
10 REM  METHODE DE NEWTON
15 LPRINT " "
20 INPUT "X INITIAL";X
25 LPRINT "X INITIAL ";X
30 INPUT "ERREUR MAXI";EM
35 LPRINT "ERREUR MAXI ";EM
36 LPRINT "EQUATION NON LINEAIRE: X-2.71828*SIN(X)=0"
40 FOR K=1 TO 50
50 GOSUB 1000
60 GOSUB 2000
70 XX=X-Y/Y1
80 ER=ABS(XX-X)
90 IF ER<EM THEN 120 ELSE X=XX
100 NEXT K
110 PRINT "NON CONVERGENCE  ERREUR=";ER:END
120 LPRINT "RACINE ";XX
125 LPRINT "NB ITERATIONS ";K
130 LPRINT "Y= ";Y:END
1000 Y=X-2.71828*SIN(X):RETURN
2000 Y1=1-2.71828*COS(X):RETURN
```

```
X INITIAL  1.5
ERREUR MAXI  .1
EQUATION NON LINEAIRE: X-2.71828*SIN(X)=0
RACINE  2.2023
NB ITERATIONS  3
Y=  .248157

X INITIAL  1.5
ERREUR MAXI  .01
EQUATION NON LINEAIRE: X-2.71828*SIN(X)=0
RACINE  2.19913
NB ITERATIONS  4
Y=  8.26764E-03

X INITIAL  1.5
ERREUR MAXI  1E-03
EQUATION NON LINEAIRE: X-2.71828*SIN(X)=0
RACINE  2.19912
NB ITERATIONS  5
Y=  1.09673E-05

X INITIAL  1.5
ERREUR MAXI  1E-04
EQUATION NON LINEAIRE: X-2.71828*SIN(X)=0
RACINE  2.19912
NB ITERATIONS  5
Y=  1.09673E-05
```

COMPLEMENTS THEORIQUES

Les méthodes itératives peuvent se mettre sous la forme générale :

$$\begin{cases} x_0 \text{ donné} \\ x_{i+1} = g(x_i) \end{cases}$$

Il faut choisir la fonction g(x) pour que la suite $x_0, x_1, x_2 \ldots x_i, x_{i+1} \ldots$ tende vers la racine α lorsque i croît indéfiniment : x_i doit converger vers α, racine de l'équation f(x)=0.

Cela signifie que la solution de g(x)=x dans l'intervalle [a,b] doit être la même que celle de f(x)=0 dans le même intervalle.

Une application (fonction) g est dite contractante s'il existe une constante L, telle que pour n'importe quelles valeurs x et y prises dans l'intervalle [a,b] :

$$|g(x) - g(y)| \leqslant L\,|x-y| \quad \text{avec } 0 \leqslant L < 1$$

Si la fonction g est dérivable, il suffit, pour que g soit contractante que :

$$|g'(x)| < 1$$

pour n'importe quel x pris dans l'intervalle $[a,b]$.

Enfin, on montre que, toujours dans l'intervalle $[a,b]$: si g est continue, contractante et reste dans l'intervalle $[a,b]$ alors, quel que soit x_0 appartenant à $[a,b]$, la suite x_i converge vers l'unique solution α de $x=g(x)$ ($\alpha \in [a,b]$).

APPLICATION A LA METHODE DE LAGRANGE

g(x) peut se mettre sous la forme :

$$g(x) = \frac{af(x)-xf(a)}{f(x)-f(a)}$$

si f est dérivable :

$$g'(x) = \frac{[af'(x)-f(a)]\,[f(x)-f(a)] - f'(x)\,[af(x)-xf(a)]}{[f(x)-f(a)]^2}$$

comme $f(\alpha)=0$, puisque α est la racine :

$$g'(\alpha) = \frac{f(a)+(\alpha-a)f'(\alpha)}{f(a)}$$

ce qui peut se mettre sous la forme (✱) :

$$g'(\alpha) = \frac{(a-\alpha)^2\, f''(c)}{2\, f(a)} \qquad a \leqslant c \leqslant \alpha$$

la convergence a lieu quand $g'(\alpha) < 1$, (il faut $f''(c).f(a) > 0$, $f(a) \neq 0$).

APPLICATION A LA METHODE DE NEWTON

Si $f''(\alpha)$ existe : $g'(\alpha) = \frac{f(\alpha).f''(\alpha)}{[f'(\alpha)]^2}$

comme $f(\alpha)=0$, $g'(\alpha)=0$, donc g(x) est toujours contractante pour les racines simples ($f'(\alpha) \neq 0$).

(✱) en utilisant la formule de Taylor :

$$f(a) = f(\alpha)+(a-\alpha)f'(\alpha)+\frac{(a-\alpha)^2}{2} f''(c) \text{ avec } a \leqslant c \leqslant \alpha$$

SYSTEMES D'EQUATIONS NON LINEAIRES

La méthode de Newton peut être généralisée pour résoudre des systèmes d'équations non linéaires de la forme :

$$\begin{cases} f_1(x_1,x_2,x_3 \ \ldots \ x_n)=0 \\ f_2(x_1,x_2,x_3 \ \ldots \ x_n)=0 \\ f_3(x_1,x_2,x_3 \ \ldots \ x_n)=0 \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ f_n(x_1,x_2,x_3 \ \ldots \ x_n)=0 \end{cases}$$

ou encore :

$$F(X)=0$$

avec :

$$X=(x_1,x_2,x_3 \ \ldots \ x_n)$$

Pour une équation à une variable, la formule de Newton est :

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

ou encore :

$$f'(x_i)\,\Delta x = -f(x_i)$$

avec :

$$\Delta x = x_{i+1}-x_i.$$

Pour un système linéaire, la formule itérative est très proche :

$$J(X_i)\ \Delta X = -F(X_i)$$

où $J(X_i)$ est le "Jacobien" : matrice d'ordre n (n correspondant au nombre de composantes de X), telle que :

$$J(X_i)=\begin{vmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \ldots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \ldots\ldots\ldots & & \\ \vdots & & & & \\ \frac{\partial f_n}{\partial x_1} & \ldots\ldots\ldots\ldots & & & \frac{\partial f_n}{\partial x_n} \end{vmatrix}$$

et $\Delta X = X_{i+1}-X_i$.

La méthode de Newton linéarise ainsi le système et il faut alors, à chaque itération, résoudre un système linéaire, (dont les inconnues sont Δx_i), défini par la formule itérative, jusqu'à ce que le vecteur $X(x_1,x_2...x_n)$ soit suffisamment proche de la solution.

Pour les systèmes à une seule variable, il suffisait de définir l'erreur admissible ε entre les résultats de deux itérations successives. Lorsqu'il s'agit d'une suite de vecteur, il faut généraliser la notion de distance : on utilise les normes.

Une norme notée $\|X\|$ possède les trois propriétés de la distance :

$$\|X\| = 0 \iff X=0$$
$$\|\lambda X\| = |\lambda| \, \|X\|$$
$$\|X+Y\| \leqslant \|X\| + \|Y\|$$

Ainsi, la suite X_i converge vers le vecteur A si :

$$\|X_i - A\| \longrightarrow 0 \text{ quand } i \longrightarrow \infty$$

En calcul numérique, on utilise fréquemment trois normes :

1 - norme euclidienne :

$$\|X\| = \sqrt{x_1^2+x_2^2+...+x_n^2}$$

2 - $$\|X\| = \Sigma|x_i|$$

3 - $$\|X\| = \max_i |x_i|$$

Exercice : *Ecrire les programmes pour le calcul de ces trois normes.*

Le programme qui est présenté ci-après ne permet de résoudre qu'un système à deux équations. La généralisation peut être faite en le complétant par un sous-programme pour résoudre à chaque itération le système d'équations linéaires.

Les calculs théoriques semblent un peu compliqués, mais l'exemple suivant va montrer comment on peut trouver facilement les coordonnées du point d'intersection d'un quart de cercle avec un quart d'ellipse : pour cela, il faut résoudre :

$$\begin{cases} x^2+y^2-9=0 & \text{(équation du cercle)} \\ \frac{x^2}{16}+\frac{y^2}{4}-1=0 & \text{(équation de l'ellipse)} \end{cases}$$

Ce système est de la forme :

$$\begin{cases} f(x,y)=0 \\ g(x,y)=0 \end{cases}$$

la formule itérative s'écrit alors :

$$\begin{vmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{vmatrix} \begin{vmatrix} \Delta x \\ \Delta y \end{vmatrix} = \begin{vmatrix} -f \\ -g \end{vmatrix}$$

ou d'une manière plus classique :

$$\frac{\partial f}{\partial x} \cdot \Delta x + \frac{\partial f}{\partial y} \cdot \Delta y = -f(x_i, y_i)$$

$$\frac{\partial g}{\partial x} \cdot \Delta x + \frac{\partial g}{\partial y} \cdot \Delta y = -g(x_i, y_i)$$

il faut rechercher les inconnues Δx et Δy ; les coefficients $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, $\frac{\partial g}{\partial x}$, $\frac{\partial g}{\partial y}$ sont les valeurs numériques prises par les dérivées partielles au point x_i, y_i. Pour un système linéaire à deux équations, la formule de Cramer est utilisable et donne directement la solution :

$$\Delta x = \frac{-f \frac{\partial g}{\partial y} + g \frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x} \cdot \frac{\partial g}{\partial y} - \frac{\partial f}{\partial y} \frac{\partial g}{\partial x}}$$

$$\Delta y = \frac{-g \frac{\partial f}{\partial x} + f \frac{\partial g}{\partial x}}{\frac{\partial f}{\partial x} \cdot \frac{\partial g}{\partial y} - \frac{\partial f}{\partial y} \frac{\partial g}{\partial x}}$$

Dans le cas présent, les dérivées partielles se calculent facilement :

$$\frac{\partial f}{\partial x} = \frac{\partial}{\partial x}(x^2+y^2-9) = 2x$$

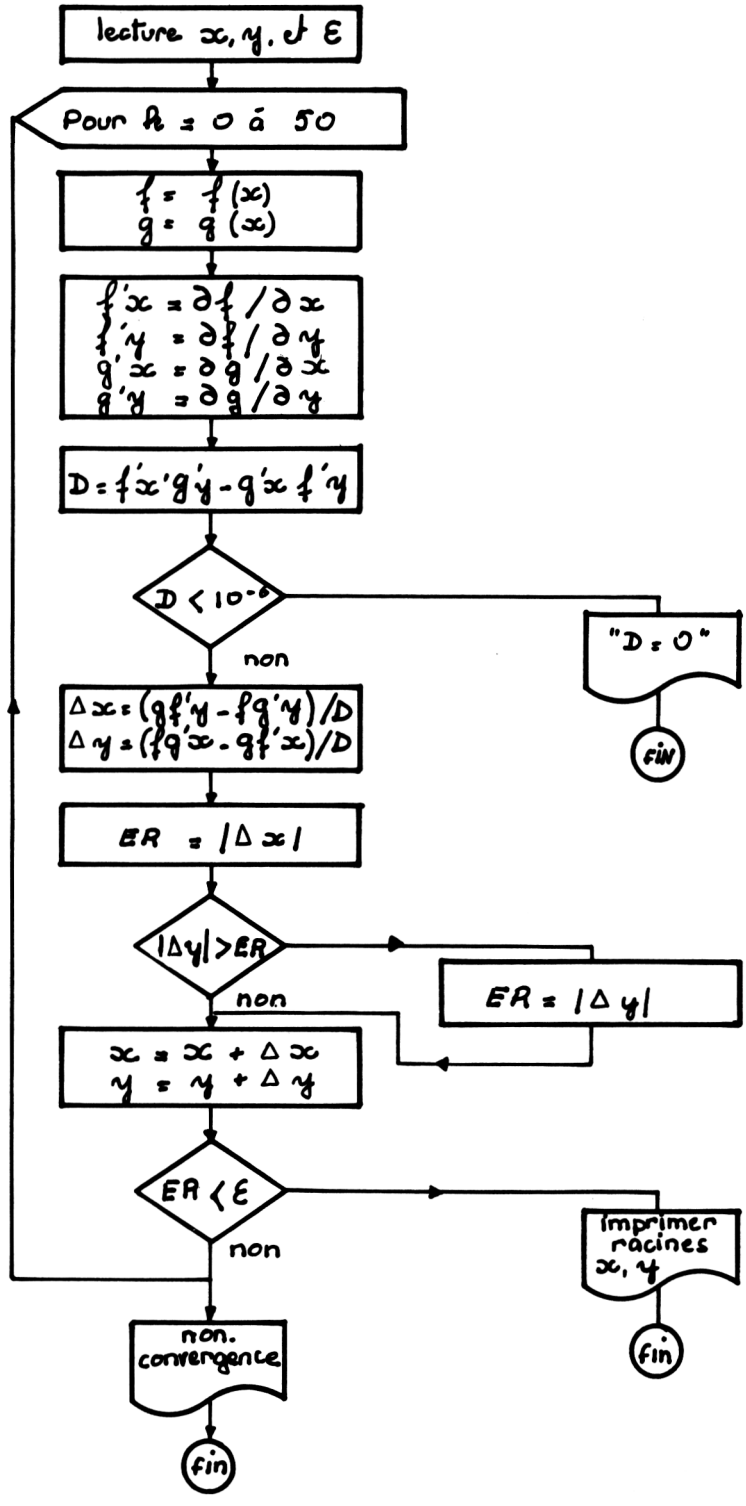
$$\frac{\partial f}{\partial y} = \frac{\partial}{\partial y}(x^2+y^2-9) = 2y$$

$$\frac{\partial g}{\partial x} \quad \frac{\partial}{\partial x}\left(\frac{x^2+y^2}{16}\frac{}{4}-1\right) = \frac{2x}{16} = \frac{x}{8}$$

$$\frac{\partial g}{\partial y} = \frac{\partial}{\partial y}\left(\frac{x^2+y^2}{16}\frac{}{4}-1\right) = \frac{2y}{4} = \frac{y}{2}$$

Les fonctions sont symbolisées par F et G (valeurs prises pour $x=x_i$ et $y=y_i$) et les dérivées partielles par FX, FY, GX, GY.

L'organigramme suivant donne le calcul itératif de la solution ; le calcul s'arrête quand max ($|x_{i+1}-x_i|$, $|y_{i+1}-y_i|$) est inférieur à ε.


lecture x, y, et ε
Pour k = 0 á 50
f = f(x)
g = g(x)
f'x = ∂f/∂x
f'y = ∂f/∂y
g'x = ∂g/∂x
g'y = ∂g/∂y
D = f'x·g'y - g'x·f'y
D < 10⁻⁶
"D = 0"
FIN
non
Δx = (gf'y - fg'y)/D
Δy = (fg'x - gf'x)/D
ER = |Δx|
|Δy| > ER
ER = |Δy|
non
x = x + Δx
y = y + Δy
ER < ε
Imprimer racines x, y
fin
non
non. convergence
fin

```
5 REM  METHODE DE NEWTON POUR SYSTEME NON LINEAIRE
6 LPRINT" "
10 INPUT "POINT INITIAL X0,Y0";X,Y
15 LPRINT "POINT INITIAL X0,Y0 ";X;Y
20 INPUT "ERREUR MAXI";EM
25 LPRINT "ERREUR MAXI ";EM
30 FOR K=1 TO 40
40 GOSUB 1000
50 GOSUB 2000
60 D=FX*GY-GX*FY
70 IF D<1E-8 THEN PRINT "ERREUR D=0":END
80 AX=(G*FY-F*GY)/D
90 AY=(F*GX-G*FX)/D
100 ER=ABS(AX):IF ABS(AY)>ER THEN ER=ABS(AY)
110 X=X+AX:Y=Y+AY
120 IF ER<EM THEN 150
130 NEXT K
140 PRINT "NON CONVERGENCE  ERREUR =";ER:END
150 LPRINT "SOLUTION DU SYSTEME"
160 LPRINT "X=";X:LPRINT "Y=";Y
165 LPRINT "NB ITERATIONS ";K
166 LPRINT "F(";X;",";Y;")=";F
167 LPRINT "G(";X;",";Y;")=";G
168 END
170 REM SYSTEME NON LINEAIRE F(X,Y) ET G(X,Y)
1000 F=X*X+Y*Y-9:FX=2*X:FY=2*Y:RETURN
2000 G=X*X/16+Y*Y/4-1:GX=X/8:GY=Y/2:RETURN
```

```
POINT INITIAL X0,Y0   1   1
ERREUR MAXI   .1
SOLUTION DU SYSTEME
X= 2.582
Y= 1.52753
NB ITERATIONS  4
F( 2.582 , 1.52753 )= .0387468
G( 2.582 , 1.52753 )= 2.42794E-03

POINT INITIAL X0,Y0   1   1
ERREUR MAXI   .01
SOLUTION DU SYSTEME
X= 2.582
Y= 1.52753
NB ITERATIONS  4
F( 2.582 , 1.52753 )= .0387468
G( 2.582 , 1.52753 )= 2.42794E-03
```

```
POINT INITIAL X0,Y0  1  1
ERREUR MAXI  1E-03
SOLUTION DU SYSTEME
X= 2.58199
Y= 1.52753
NB ITERATIONS  5
F( 2.58199 , 1.52753 )= 5.62668E-05
G( 2.58199 , 1.52753 )= 3.45707E-06

POINT INITIAL X0,Y0  1  1
ERREUR MAXI  1E-04
SOLUTION DU SYSTEME
X= 2.58199
Y= 1.52753
NB ITERATIONS  5
F( 2.58199 , 1.52753 )= 5.62668E-05
G( 2.58199 , 1.52753 )= 3.45707E-06

POINT INITIAL X0,Y0  1  1
ERREUR MAXI  1E-05
SOLUTION DU SYSTEME
X= 2.58199
Y= 1.52753
NB ITERATIONS  6
F( 2.58199 , 1.52753 )= 0
G( 2.58199 , 1.52753 )= 0
```

POLYNOMES

INTRODUCTION

Les polynômes interviennent souvent dans les calculs numériques, notamment pour le calcul des solutions d'équations différentielles linéaires, pour le calcul des valeurs propres des matrices, pour les approximations ...

La forme générale est la suivante :

$$P_n(x) = a_0x^n+a_1x^{n-1}+ \ldots +a_{n-1}x+a_n$$

les coefficients $a_0, a_1 \ldots a_n$ sont des constantes réelles ou complexes. Le degré du polynôme est n.

Avant de commencer les calculs, il faut rappeler quelques définitions et propriétés (sans donner les démonstrations).

DIVISION

Soient un polynôme de degré n, $P_n(x)$ et un polynôme de degré $n' \leq n$, $F_{n'}(x)$, alors $Q(x)$ et $R(x)$ sont respectivement quotient et reste de la division de $P_n(x)$ par $F_{n'}(x)$, tels que :

$$P_n(x) = Q_{n''}(x) \cdot F_{n'}(x) + R(x)$$

avec : n"=n-n' et degré de R < degré de F.

Le reste de la division de $P_n(x)$ par $(x-\alpha)$ est la valeur prise par ce polynôme pour $x=\alpha$: $R=P_n(\alpha)$; en effet, par définition, quel que soit x :

$$P_n(x) = (x-\alpha)\ Q_{n-1}\ (x) + R_0$$

si $x = \alpha$

$$P_n(\alpha) = (\alpha-\alpha)\ Q_{n-1}\ (x) + R_0$$
$$= R_0$$

RACINES

Un polynôme $P_n(x)$ possède n racines réelles ou complexes :

- la somme des racines $\sum_i x_i$ est égale à $-a_1/a_0$
- le produit des racines $\prod_i x_i$ est égal à $(-1)^n \frac{a_n}{a_0}$
- si un polynôme de la forme $P_n(x) = x^n+a_1x^{n-1}+ \ldots + a_{n-1}x+a_n$ a n racines distinctes, il peut s'écrire sous la forme :

$$P_n(x) = (x-\alpha_1)(x-\alpha_2) \ldots (x-\alpha_n)$$

Les opérations classiques sont :

- évaluation pour $x=\alpha$
- division par un binôme $(x-\alpha)$
- produit par un binôme $(x-\alpha)$
- dérivée
- recherche des racines

OPERATIONS POLYNOMIALES

EVALUATION D'UN POLYNOME ($x=\alpha$)

Pour calculer $P_n(\alpha)$, il suffit de reprendre l'expression de P_n donnée au paragraphe précédent :

$$P_n(x) = a_0x^n+a_1x^{n-1}+ \ldots +a_{n-1}x+a_n$$

et faire les calculs en remplaçant x par α.

Mais l'algorithme suivant, de Hörner, est nettement plus efficace (plus précis et plus rapide): l'expression du polynôme est transformée en mettant x en facteur :

$$P_n(x) = x\ (x\ (x\ (x\ \ldots\ a_{n-3})+a_{n-2})+a_{n-1})+a_n$$

ou sous forme récurrente :

$$P_0 = a_0$$
$$P_1 = a_0x+a_1$$
$$\text{----------}$$
$$P_k = P_{k-i}x+a_k$$

PRODUIT PAR UN BINOME $(x-\alpha)$

Les coefficients du polynôme $H_{n+1}(x)=P_n(x)\,(x-\alpha)$ sont calculés directement en développant :

$$(a_0x^n+a_1x^{n-1}+\ \ldots\ +a_{n-1}x+a_n)\ \ (x-\alpha)=a_0x^{n+1}+(a_1-\alpha a_0)x^n$$
$$+\ (a_2-\alpha a_1)x^{n-1}+\ \ldots$$
$$= h_0x^{n+1}+h_1x^n+\ \ldots\ +h_nx+h_{n+1}$$

avec $h_0=a_0$, $h_1=a_1-\alpha a_0$ ...

$$h_k=a_k-\alpha\, a_{k-1}$$

DIVISION PAR $(x-\alpha)$

On a vu que $P_n(x)= (x-\alpha)\ Q_{n-1}+R_0$; en développant : (si $a_0=1$).

$$x^n+a_1x^{n-1}+a_2x^{n-2}+\ \ldots\ +a_{n-1}x+a_n=$$
$$(x-\alpha)\,(x^{n-1}+q_1x^{n-2}+q_2x^{n-3}+\ \ldots\ +q_{n-2}x+q_{n-1})+q_n$$

en effectuant :

$$x^n+a_1x^{n-1}+a_2x^{n-2}+\ \ldots\ +a_{n-1}x+a_n=$$
$$x^n+(q_1-\alpha)x^{n-1}+(q_2-\alpha q_1)x^{n-2}+(q_3-\alpha q_2)x^{n-3}+\ \ldots$$
$$+(q_{n-1}-q_{n-2}\alpha)x+(q_n-\alpha q_{n-1})$$

en identifiant les termes de même degré :

$$a_1=q_1-\alpha\ ,\ a_2=q_2-\alpha q_1\ ,\ a_3=q_3-\alpha q_2\ \ldots$$

d'où :

$$q_1 = a_1+\alpha$$
$$q_2 = a_2+\alpha\, q_1$$
$$q_3 = a_3+\alpha\, q_2$$
$$\text{----------}$$
$$q_k = a_k+\alpha\, q_{k-1}$$

DERIVEE D'UN POLYNOME : $P'_n(x)$

Il suffit de dériver chaque terme :

si $P_n(x) = a_0x^n+a_1x^{n-1}+ \ldots +a_{n-1}x+a_n$, alors :

$$P'_n(x) = na_0x^{n-1}+(n-1)a_1x^{n-2}+ \ldots +2a_{n-2}x+a_{n-1}$$

Pour la méthode de Birge Vieta, exposée au paragraphe suivant, la dérivée est calculée par récurrence : la division de $P_n(x)$ par $(x-\alpha)$ donne : $P_n(x) = (x-\alpha)P_{n-1}(x) +p_n$.

On a vu que $p_n = P_n(\alpha)$, en dérivant l'expression ci-dessus :

$$P'_n(x) = (x-\alpha)P'_{n-1}(x)+P_{n-1}(x)$$

quand $x = \alpha$

$$P'_n(\alpha) = P_{n-1}(\alpha)$$

Au lieu de faire cette dérivation pour obtenir $P'_n(\alpha)$, on peut diviser $P_{n-1}(x)$ par $(x-\alpha)$:

$$P_{n-1}(x) = (x-\alpha)P_{n-2}(x)+q_n$$

On a vu que $q_n = P_{n-1}(\alpha)$, donc on obtient ainsi $P_n'(\alpha) = q_n$.

La division est effectuée comme précédemment en utilisant les formules récurrentes dont la programmation est simple.

RACINES D'UN TRINOME

La méthode de calcul des racines d'une équation du second degré est classique, mais elle sera utilisée sous la forme générale par l'algorithme de Bairstow.

Le trinôme du second degré s'écrit :

$$ax^2+bx+c$$

les racines sont x_1 et x_2 ; elles peuvent être réelles, doubles ou complexes conjuguées suivant le signe du discriminant Δ :

$$\Delta = b^2-4ac$$

. si $\Delta < 0$, les racines sont complexes :

$$x = \frac{-b \pm i\sqrt{-\Delta}}{2a}$$

ou encore :

$$x_1 = -\frac{b}{2a} - i\frac{\sqrt{-\Delta}}{2a}$$

$$x_2 = -\frac{b}{2a} + i\frac{\sqrt{-\Delta}}{2a}$$

si $R(x_1)$ est la partie réelle de x_1

et $I(x_1)$ est la partie imaginaire

alors $x_1.x_2 = R^2(x_1) + I^2(x_1) = R^2(x_2) + I^2(x_2)$

Remarque :

$$R(x_1) = R(x_2) = -\frac{b}{2a}$$

en effectuant, on retrouve bien $x_1.x_2 = -\frac{c}{a}$

. Si $\Delta = 0$, les racines sont réelles et doubles :

$$x1 = x2 = -\frac{b}{2a}$$

. Si $\Delta > 0$, les racines sont réelles et distinctes :

$$x_1 = \frac{-b-\sqrt{\Delta}}{2a}$$

$$x_2 = \frac{-b+\sqrt{\Delta}}{2a}$$

L'organigramme et le programme seront donnés plus loin : l'algorithme de Bairstow utilise ces formules.

METHODE ITERATIVE DE BIRGE VIETA

Les racines réelles, simples, sont calculées en utilisant la méthode de Newton :

$$x_{i+1} = x_i - \frac{P_n(x_i)}{P'_n(x_i)}$$

Pour démarrer le calcul, il faut choisir une valeur initiale x_0. On peut prendre, par exemple :

$$x_0 = -\frac{an}{an-1}$$

Ensuite, il faut évaluer $P_n(x_i)$ au moyen de l'algorithme de Hörner et $P'_n(x_i)$ en effectuant la division de $P_n(x_i)$ par $(x-x_i)$. Donc :

$$x_1 = x_0 - \frac{P_n(x_0)}{P'_n(x_0)}$$

$$x_2 = x_1 - \frac{P_n(x_1)}{P'_n(x_1)}$$

et ainsi de suite, jusqu'à ce que $|x_{i+1}-x_i| < \varepsilon$ ou bien $P_n(x_{i+1}) \simeq 0$, d'où $\alpha_1 \simeq x_{i+1}$.

La division de $P_n(x)$ par $(x-\alpha_1)$ donne $P_{n-1}(x)$ et une autre racine est recherchée de la même manière, en partant d'un nouvel x_0 choisi comme précédemment ou en prenant $x0 = \alpha_1$. Lorsque le degré est tombé à 2, les formules classiques peuvent être appliquées.

Donc, à chaque itération, $P_n(x_i)$ et $P'_n(x_i)$ doivent être évalués. Ces calculs sont effectués en utilisant les formules de la division :

$$P_n(x) = x^n+a_1x^{n-1}+a_2x^{n-2}+ \ldots +a_{n-1}x+a_n$$

$$= (x-x_i)(x^{n-1}+p_1x^{n-2}+p_2x^{n-3}+ \ldots +p_{n-2}x+p_{n-1})+p_n$$

où les coefficients p_k sont calculés par récurrence :

$$p_1 = a_1+x_i$$

$$p_2 = a_2+p_1x_i$$

$$\text{----------}$$

$$p_k = a_k+p_{k-1}x_i$$

$$\text{----------}$$

$$p_n = a_n+p_{n-1}x_i$$

d'après la propriété du reste de la division de $P_n(x)$ par $(x-x_i)$, $p_n = P_n(x_i)$, c'est la première valeur recherchée.

L'évaluation du polynôme pour $x=x_i$ au moyen de l'algorithme de Hörner se fait exactement de la même manière, mais ici, tous les coefficients sont conservés pour les calculs suivants. Il a été montré que $P'_n(x_i)$ est égal au reste de la division de $P_{n-1}(x)$ par $x-x_i$, c'est-à-dire que $P_{n-1}(x)$, connu par les coefficients p_k va subir le même traitement que $P_n(x)$:

$$P_{n-1}(x) = x^{n-1}+p_1x^{n-2}+p_2x^{n-3}+ \ldots +p_{n-2}x+p_{n-1})+p_n$$

$$= (x-x_i)(q_1x^{n-2}+q_2x^{n-3}+ \ldots +q_{n-2}x+q_{n-1})+q_n$$

avec : $q_1 = 1$, $q_2 = p_1+q_1x_i$

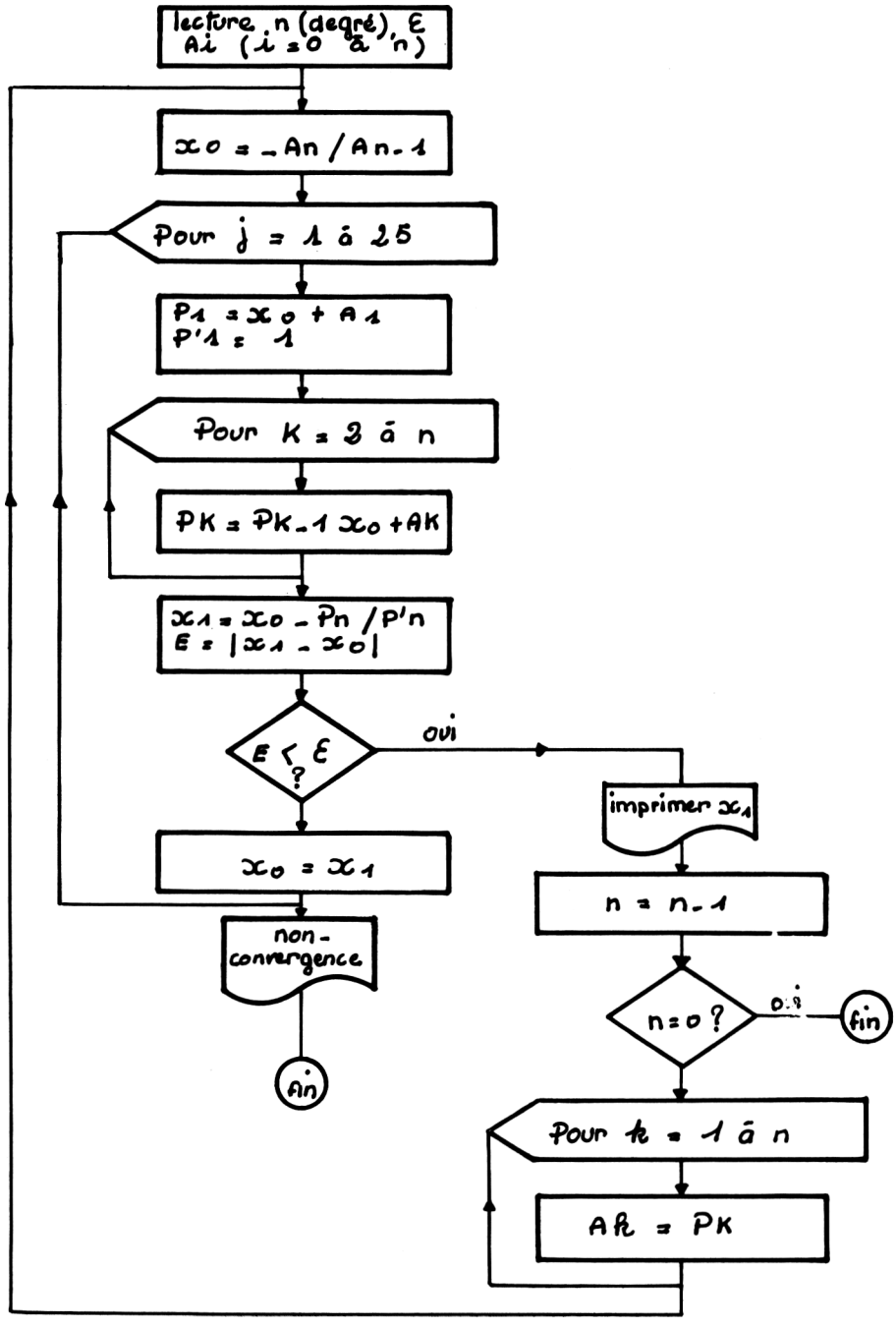
$$q_k = p_{k-1} + q_{k-1} x_i$$
$$q_n = p_{n-1} + q_{n-1} x_i$$

Avec les notations du développement ci-dessus, q_n est le reste de la division de $P_{n-1}(x)$ par $x-x_i$, donc $q_n = P_{n-1}(x_i)$; c'est la dérivée recherchée puisque $P'_n(x_i) = P_{n-1}(x_i)$.

A partir de $P_n(x_i)$ et $P'_n(x_i)$, la formule de Newton va donner x_{i+1} :

$$x_{i+1} = x_i - \frac{P_n(x_i)}{P'_n(x_i)}$$

si la précision requise n'est pas atteinte, les calculs sont refaits avec x_{i+1} ($P_n(x_{i+1})$ et $P'_n(x_{i+1})$) pour obtenir x_{i+2} et ainsi de suite. Lorsque l'écart $|x_{k+1}-x_k|$ est admissible, x_{k+1} est considéré comme racine ; pour rechercher les autres racines, il suffit de diviser $P_n(x)$ par $(x-x_{k+1})$ et de recommencer tous les calculs avec $P_{n-1}(x)$; il n'est pas nécessaire d'effectuer cette division puisqu'à chaque itération, les coefficients p_k du polynôme $P_{n-1}(x)$ sont évalués et conservés.


lecture n (degré), ε
Ai (i = 0 à n)
x0 = - An / An-1
Pour j = 1 à 25
P1 = x0 + A1
P'1 = 1
Pour K = 2 à n
PK = PK-1 x0 + AK
x1 = x0 - Pn / P'n
E = |x1 - x0|
E < ε ?
oui
imprimer x1
x0 = x1
n = n-1
non-convergence
n = 0 ?
oui
fin
An
Pour k = 1 à n
Ak = PK

```
5 REM RACINES D'UN POLYNOME * METHODE DE BIRGE VIETA
10 READ ND,EM
20 DIM A(ND),P(ND),DP(ND)
30 FOR K=0 TO ND:READ A(K):NEXT K
35 LPRINT "COEFFICIENTS DU POLYNOME"
40 FOR K=0 TO ND:LPRINT "A( ";K;")=";A(K):NEXT K
50 N=ND
70 X0=-A(N)/A(N-1)
80 FOR J=1 TO 25
90 REM  CALCUL COEFFICIENTS DE P ET DP
100 P(1)=X0+A(1)
110 DP(1)=1
120 FOR K=2 TO N
130 P(K)=P(K-1)*X0+A(K)
140 DP(K)=DP(K-1)*X0+P(K-1)
150 NEXT K
160 X1=X0-P(N)/DP(N)
170 ER=ABS(X1-X0)
180 IF ER<EM THEN 210 ELSE X0=X1
190 NEXT J
200 PRINT "NON CONVERGENCE  ERREUR=";ER:END
210 LPRINT "RACINE",X1
220 N=N-1:IF N=0 THEN END
230 FOR K=1 TO N
240 A(K)=P(K)
250 NEXT K
260 GOTO 70
3000 DATA 4,0.00001
3001 DATA 1,2,-13,-14,24
```

```
COEFFICIENTS DU POLYNOME
A(  0 )= 1
A(  1 )= 2
A(  2 )=-13
A(  3 )=-14
A(  4 )= 24
RACINE            1
RACINE           -2
RACINE            3
RACINE           -4
```

METHODE ITERATIVE DE BAIRSTOW

Le principe de cet algorithme consiste à extraire du polynôme $P_n(x)$, un trinôme du second degré $Q_2(x) = x^2-sx+p$ dont on peut calculer très facilement les racines réelles ou complexes au moyen des formules classiques.

La division du polynôme par le trinôme conduit à l'identité :

$$P_n(x) = Q_2(x).P_{n-2}(x)+R_1(x)$$

avec :
$$P_n(x) = a_0x^4+a_1x^{n-1}+ \dots +a_{n-1}x+a_n$$
$$Q_2(x) = x^2-sx+p$$
$$P_{n-2}(x) = b_0x^{n-2}+b_1x^{n-3} \dots +b_{n-3}x+b_{n-2}$$
$$R_1(x) = \alpha x+\beta$$

Pour obtenir un quotient exact, il faut rechercher les valeurs des paramètres s et p qui annulent $R_1(x)$ (qui annulent α et β).

Pour s et p donnés, les coefficients b du polynôme $P_{n-2}(x)$ et les coefficients α et β se calculent par récurrence. Les formules sont obtenues en développant l'expression $P_n = Q_2.P_{n-2}+R_1$, en effectuant tous les calculs et en identifiant les coefficients des termes de même degré :

$$a_0x^n+a_1x^{n-1}+a_2x^{n-2}+ \dots +a_{n-1}x+a_n =$$
$$(x^2-sx+p)(b_0x^{n-2}+b_1x^{n-3}+ \dots +b_{n-3}x+b_{n-2})+\alpha x+\beta$$

termes en	coefficients de P_n	coefficients de $Q.P_{n-2}+R$
x^n	a_0	b_0
x^{k-1}	a_1	b_1-sb_0
x^{k-2}	a_2	$b_2-sb_1+pb_0$
⋮	⋮	⋮
x^{n-k}	a_k	$b_k-sb_{k-1}+pb_{k-2}$
⋮	⋮	⋮
x	a_{n-1}	$\alpha-sb_{n-2}+pb_{n-3}$
x^0	a_n	$\beta+pb_{n-2}$

d'où :

$$b_0 = a_0$$
$$b_1 = a_1+sb_0$$
$$b_2 = a_2+sb_1-pb_0$$
$$\dots\dots\dots\dots\dots$$
$$b_k = a_k+sb_{k-1}-pb_{k-2}$$
$$\dots\dots\dots\dots\dots$$
$$\alpha = a_{n-1}+sb_{n-2}-pb_{n-3}$$
$$\beta = a_n-pb_{n-2}$$

On remarque que α est calculé à partir de la même formule de récurrence que les coefficients b_k et correspond alors au coefficient b_{n-1} :

$$b_{n-1}=a_{n-1}+sb_{n-2}-pb_{n-3} = \alpha$$

en posant $\beta = b_n - sb_{n-1}$, la même formule peut encore être utilisée :

$$b_n = a_n+sb_{n-1}-pb_{n-2}$$

donc, finalement $R_1(x) = \alpha x+\beta$

$$= b_{n-1}(x-s)+b_n$$

Evidemment, les coefficients b dépendent de s et p et il faut maintenant rechercher les valeurs particulières s^* et p^* qui annulent b_{n-1} et b_n ; dans ce cas, $R_1(x) = 0$ et les racines de $x^2-s^*x+p^*$ sont racines du polynôme $P_n(x)$.

b_{n-1} et b_n sont des fonctions de s et p :

$$b_{n-1} = f(s,p)$$

$$b_n = g(s,p)$$

Trouver s^* et p^* c'est résoudre le système non linéaire :

$$\begin{cases} f(s,p) = 0 \\ g(s,p) = 0 \end{cases}$$

cela se fait facilement en utilisant la méthode de Newton (voir chapitre 1).

Ainsi, pour une équation non linéaire à une seule variable, la formule itérative est :

$x_{i+1} = x_i - \frac{f(x)}{f'(x)}$ ou encore $f'(x_i)(x_{i+1}-x_i) = -f(x_i)$

A deux dimensions, la formule devient :

$$J(X_i)(X_{i+1}-X_i) = -F(X_i)$$

avec : $X_i = \begin{pmatrix} s_i \\ p_i \end{pmatrix}$, $X_{i+1} = \begin{pmatrix} s_{i+1} \\ p_{i+1} \end{pmatrix}$

$$F(X_i) = \begin{vmatrix} f(s_i,p_i) \\ g(s_i,p_i) \end{vmatrix}$$

et $$J(X_i) = \begin{vmatrix} \frac{\partial f}{\partial s} & \frac{\partial f}{\partial p} \\ \frac{\partial g}{\partial s} & \frac{\partial g}{\partial p} \end{vmatrix}$$ pour $s = s_i$ et $p = p_i$

$$J(X_i)\,\Delta X = -F(X_i) \quad \text{avec} \quad \Delta X = \begin{pmatrix} s_{i+1}-s_i \\ p_{i+1}-p_i \end{pmatrix}$$

Cette dernière relation correspond à un système linéaire de deux équations à deux inconnues $\Delta s = s_{i+1}-s_i$ et $\Delta p = p_{i+1}-p_i$:

$$\frac{\partial f}{\partial s}\,\Delta s + \frac{\partial f}{\partial p}\,\Delta p = -f(s_i,p_i)$$

$$\frac{\partial g}{\partial s}\,\Delta s + \frac{\partial g}{\partial p}\,\Delta p = -g(s_i,p_i)$$

en utilisant la formule de Cramer :

$$\Delta s = \frac{-f\,\frac{\partial g}{\partial p} + g\,\frac{\partial f}{\partial p}}{\delta}$$

avec $\delta = \frac{\partial f}{\partial s}\,\frac{\partial g}{\partial p} - \frac{\partial f}{\partial p}\,\frac{\partial g}{\partial s}$

$$\Delta p = \frac{-g\,\frac{\partial f}{\partial s} + f\,\frac{\partial g}{\partial s}}{\delta}$$

Pour appliquer ces formules, les dérivées partielles $\frac{\partial f}{\partial s}$, $\frac{\partial f}{\partial p}$, $\frac{\partial g}{\partial s}$, $\frac{\partial g}{\partial p}$ doivent être évaluées.

Ces calculs se font encore par récurrence :

$b_0 = a_0$, donc $\frac{\partial b_0}{\partial s} = 0$ et $\frac{\partial b_0}{\partial p} = 0$

$b_1 = a_1+sb_0$, d'où $\frac{\partial b_1}{\partial s} = b_0$; $\frac{\partial b_1}{\partial p} = 0$

$b_2 = a_2+sb_1-pb_0$, $\frac{\partial b_2}{\partial s} = \frac{\partial}{\partial s}(a_2) + \frac{\partial}{\partial s}(sb_1) - \frac{\partial}{\partial s}(pb_0)$

$\frac{\partial}{\partial s}\,a_2 = 0$; $\frac{\partial}{\partial s}(sb_1) = s\,\frac{\partial b_1}{\partial s} + b_1$; $\frac{\partial}{\partial s}\,pb_0 = 0$

d'où $\frac{\partial b_2}{\partial s} = b_1+sb_0$

$b_3 = a_3+sb_2-pb_1$, $\frac{\partial b_3}{\partial s} = b_2+s\,\frac{\partial b_2}{\partial s} - p\,\frac{\partial b_1}{\partial s}$

Si on pose $\frac{\partial b_k}{\partial s} = C_{k-1}$

alors :

$$c_0 = b_0$$
$$c_1 = b_1+sb_0 = b_1+sc_0$$
$$c_2 = b_2+sc_1 - pc_0$$
$$\text{----------}$$
$$c_k = b_k+sc_{k-1}-pc_{k-2}$$
$$\text{----------}$$

enfin : $c_{n-1} = \frac{\partial b_n}{\partial s} = b_{n-1}+sc_{n-2}-pc_{n-3}$

On voit que les coefficients c_k se calculent de la même manière que les coefficients b_k.

Exercice : *effectuer les mêmes calculs avec p, c'est-à-dire, calculer* $\frac{\partial b_k}{\partial p}$ *; montrer qu'en posant* $d_k = -\frac{\partial b_{k-2}}{\partial p}$ *, on a* $d_k = c_k$ *(k = 0 à n-2).*

Finalement, comme f : b_{n-1} et g = b_n :

$\frac{\partial f}{\partial s} = c_{n-2}$

$\frac{\partial f}{\partial p} = c_{n-3}$

$\frac{\partial g}{\partial s} = c_{n-1}$

$\frac{\partial g}{\partial p} = c_{n-2}$

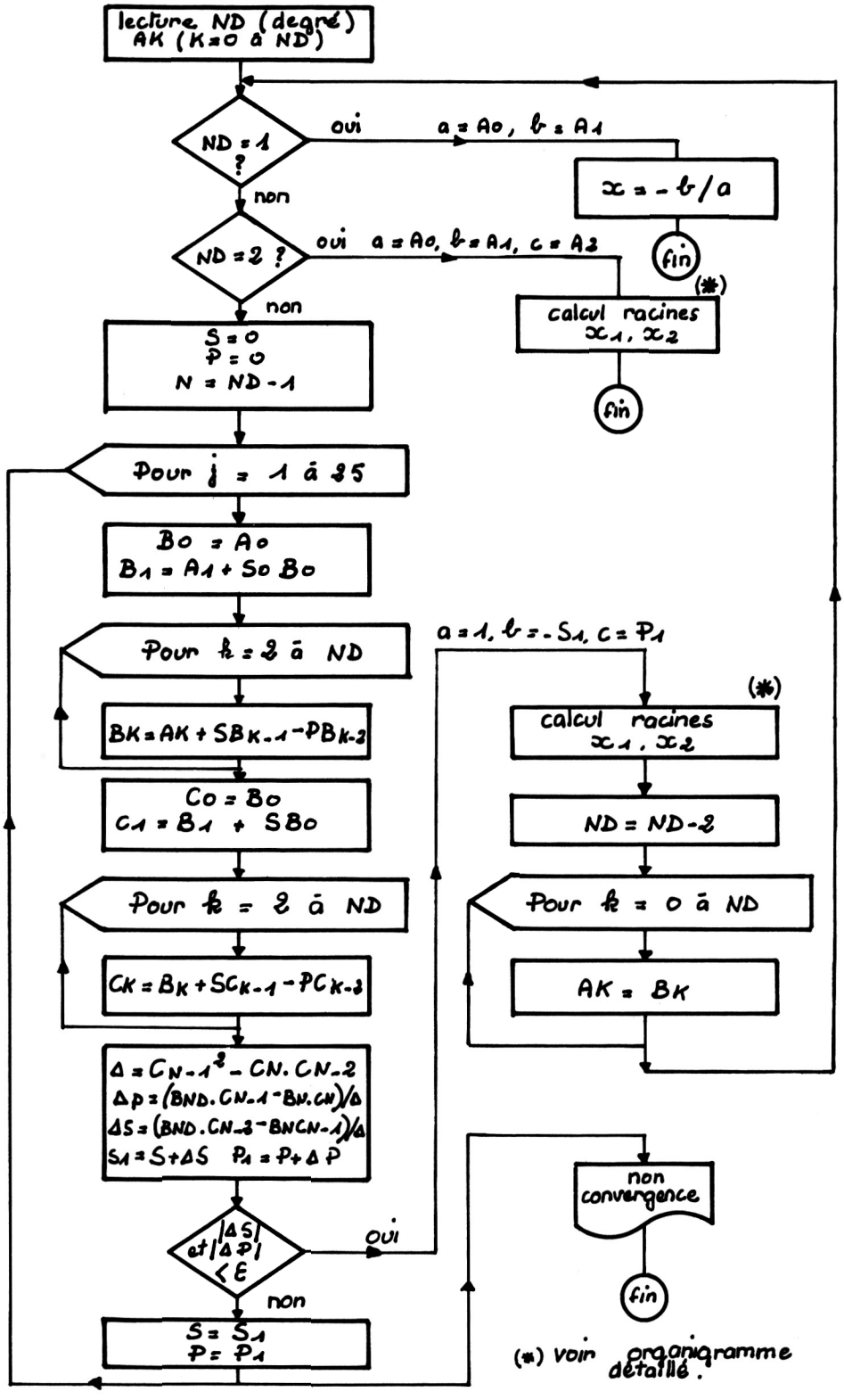
d'où :

$$\Delta s = \frac{b_{n-1}\cdot c_{n-2}-b_n c_{n-3}}{c_{n-1}\cdot c_{n-3}-c_{n-2}^2}$$

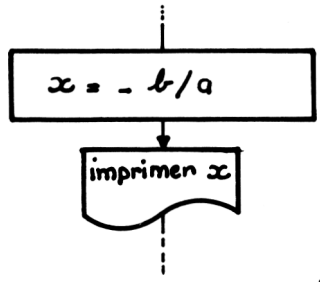
$$\Delta p = \frac{b_{n-1}\cdot c_{n-1}-b_n c_{n-2}}{c_{n-1}\cdot c_{n-3}-c_{n-2}^2}$$

L'organigramme résume ces calculs :

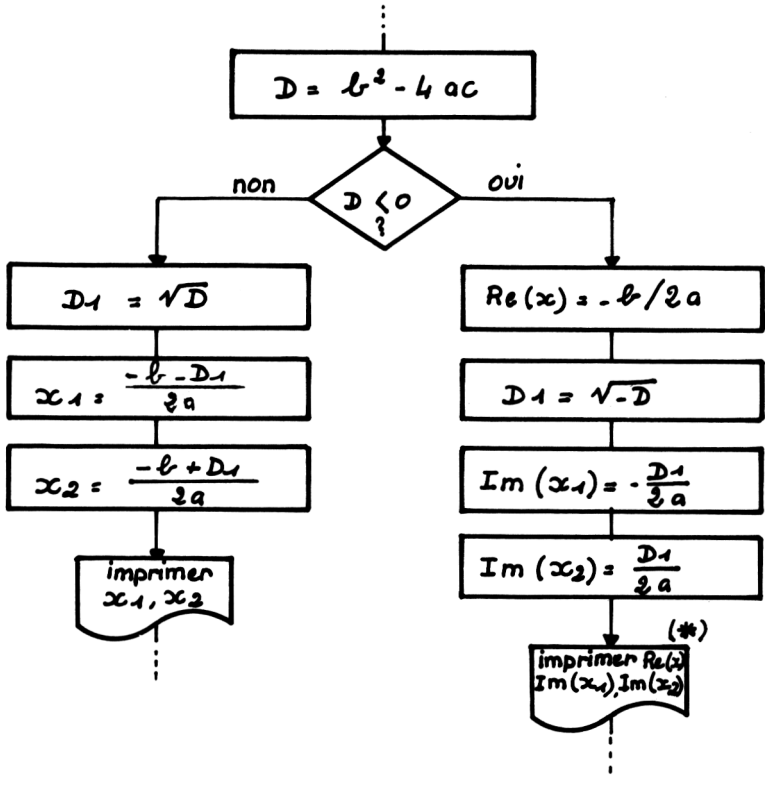
⟶


lecture ND (degré)
AK (K=0 à ND)
ND = 1 ?
oui
a = Ao, b = A1
x = - b / a
fin
non
ND = 2 ?
oui a = Ao, b = A1, c = A2
(*)
calcul racines x1, x2
fin
non
S = 0
P = 0
N = ND - 1
Pour j = 1 à 25
Bo = Ao
B1 = A1 + So Bo
Pour k = 2 à ND
BK = AK + SBK-1 - PBK-2
Co = Bo
C1 = B1 + SBo
Pour k = 2 à ND
CK = BK + SCK-1 - PCK-2
Δ = CN-1² - CN. CN-2
Δp = (BND. CN-1 - BN. CN)/Δ
ΔS = (BND. CN-2 - BNCN-1)/Δ
S1 = S + ΔS P1 = P + Δ P
|ΔS| et |ΔP| < E
oui
a = 1, b = - S1, c = P1
(*)
calcul racines x1, x2
ND = ND - 2
Pour k = 0 à ND
AK = BK
non convergence
fin
non
S = S1
P = P1
(*) voir organigramme détaillé.

Racine du polynôme de degré 1 : ax+b



Racines du polynôme de degré 2 : ax^2+bx+c



(*) $R_e(x)$ *partie réelle de x*

$I_m(x_1)$ *partie imaginaire de* x_1

```
5 REM RACINES D'UN POLYNOME * METHODE DE BAIRSTOW
10 READ ND
20 DIM A(ND),B(ND),C(ND)
30 FOR K=0 TO ND:READ A(K):NEXT K
35 LPRINT "COEFFICIENTS DU POLYNOME"
40 FOR K=0 TO ND:LPRINT "A( ";K;")=";A(K):NEXT K
45 LPRINT "***** R A C I N E S *****"
50 AX=A(0):BX=A(1):CX=A(2)
60 IF ND=1 THEN 1000 ELSE IF ND=2 THEN GOSUB 2000
65 IF ND=2 THEN END
66 REM INITIALISATION S ET P (QUELCONQUE)
70 S=1.23456:P=0.56789:N=ND-1
80 FOR J=1 TO 25
90 B(0)=A(0):B(1)=A(1)+S*B(0)
100 FOR K=2 TO ND:B(K)=A(K)+S*B(K-1)-P*B(K-2):NEXT K
110 C(0)=B(0):C(1)=B(1)+S*B(0)
120 FOR K=2 TO N:C(K)=B(K)+S*C(K-1)-P*C(K-2):NEXT K
130 DT=C(N-1)↑2-C(N)*C(N-2)
140 DP=(B(ND)*C(N-1)-B(N)*C(N))/DT
150 DS=(B(ND)*C(N-2)-B(N)*C(N-1))/DT
160 S1=S+DS:P1=P+DP
170 IF (ABS(DS)<1E-5) AND (ABS(DP)<1E-5) THEN 240
180 S=S1:P=P1:NEXT J
190 PRINT "NON CONVERGENCE",DS,DP
240 AX=1:BX=-S1:CX=P1:GOSUB 2000
260 ND=ND-2
270 FOR K=0 TO ND:A(K)=B(K):NEXT K
280 GOTO 50
1000 X1=-BX/AX:PRINT "X=";X1:END
2000 D=BX↑2-4*AX*CX
2010 AD=2*AX
2020 IF D<0 THEN 2080
2030 D1=SQR(D)
2040 X1=(-BX-D1)/AD
2050 X2=(-BX+D1)/AD
2060 LPRINT "X1=";X1:LPRINT "X2=";X2
2070 RETURN
2080 XR=-BX/AD
2090 D1=SQR(-D)
2100 X1=-D1/AD
2110 X2=D1/AD
2120 LPRINT "RACINES COMPLEXES"
2125 LPRINT "R(X1)=";XR;" IM(X1)=";X1
2130 LPRINT "R(X2)=";XR;"IM(X2)=";X2
2140 GOTO 2070
3000 DATA 4
3001 DATA1,0,0,0,-1
```

```
COEFFICIENTS DU POLYNOME
A(  0 )= 1
A(  1 )= 0
A(  2 )= 0
A(  3 )= 0
A(  4 )=-1
****  R A C I N E S ****
X1=-1
X2= 1
RACINES COMPLEXES
R(X1)= 1.1687E-10  IM(X1)=-1
R(X2)= 1.1687E-10 IM(X2)= 1

COEFFICIENTS DU POLYNOME
A(  0 )= 1
A(  1 )= 2
A(  2 )=-13
A(  3 )=-14
A(  4 )= 24
****  R A C I N E S ****
X1=-4
X2= 3
X1=-2
X2= 1
```

METHODE DE BERNOUILLI

Cette méthode permet de calculer la plus grande racine α ; en faisant la division par $(x-\alpha)$, la méthode peut être appliquée de nouveau et ainsi de suite pour déterminer toutes les racines.

On considère l'équation aux différences φ associée au polynôme $P_n(x) = a_0x^n+a_1x^{n-1}+ \ldots +a_{n-1}x+a_n$:

$$\varphi = a_0y_k+a_1y_{k-1}+ \ldots +a_ny_{k-n}=0$$

Les solutions particulières de cette équation (qui ressemble à une équation différentielle linéaire), sont de la forme $y_k=\beta^k$ où les constantes β sont les racines de l'équation caractéristique :

$$a_0\beta^k+a_1\beta^{k-1}+ \ldots +a_n\beta^{k-n}=0$$

qui est de la même forme que l'équation polynomiale $P_n(x)=0$ où les racines, supposées distinctes sont x_1, x_2, $\ldots$ x_n.

L'équation aux différences étant linéaire, la solution générale est une combinaison linéaire :

$$y_k = c_1x_1^k+c_2x_2^k+ \ldots +c_nx_n^k$$

Si les racines sont telles que :

$$|x_1| > |x_2| > |x_3| > \ldots > |x_n|$$

alors : $$y_k = c_1 x_1^k \left[1 + \frac{c_2}{c_1}\left(\frac{x_2}{x_1}\right)^k + \ldots\right]$$

et : $$y_{k+1} = c_1 x_1^{k+1} \left[1 + \frac{c_2}{c_1}\left(\frac{x_2}{x_1}\right)^{k+1} + \ldots\right]$$

d'où : $$\frac{y_{k+1}}{y_k} = x_1 \frac{\left[1 + \frac{c_2}{c_1}\left(\frac{x_2}{x_1}\right)^{k+1} + \ldots\right]}{\left[1 + \frac{c_2}{c_1}\left(\frac{x_2}{x_1}\right)^{k} + \ldots\right]}$$

comme : $x_1 > x_2$

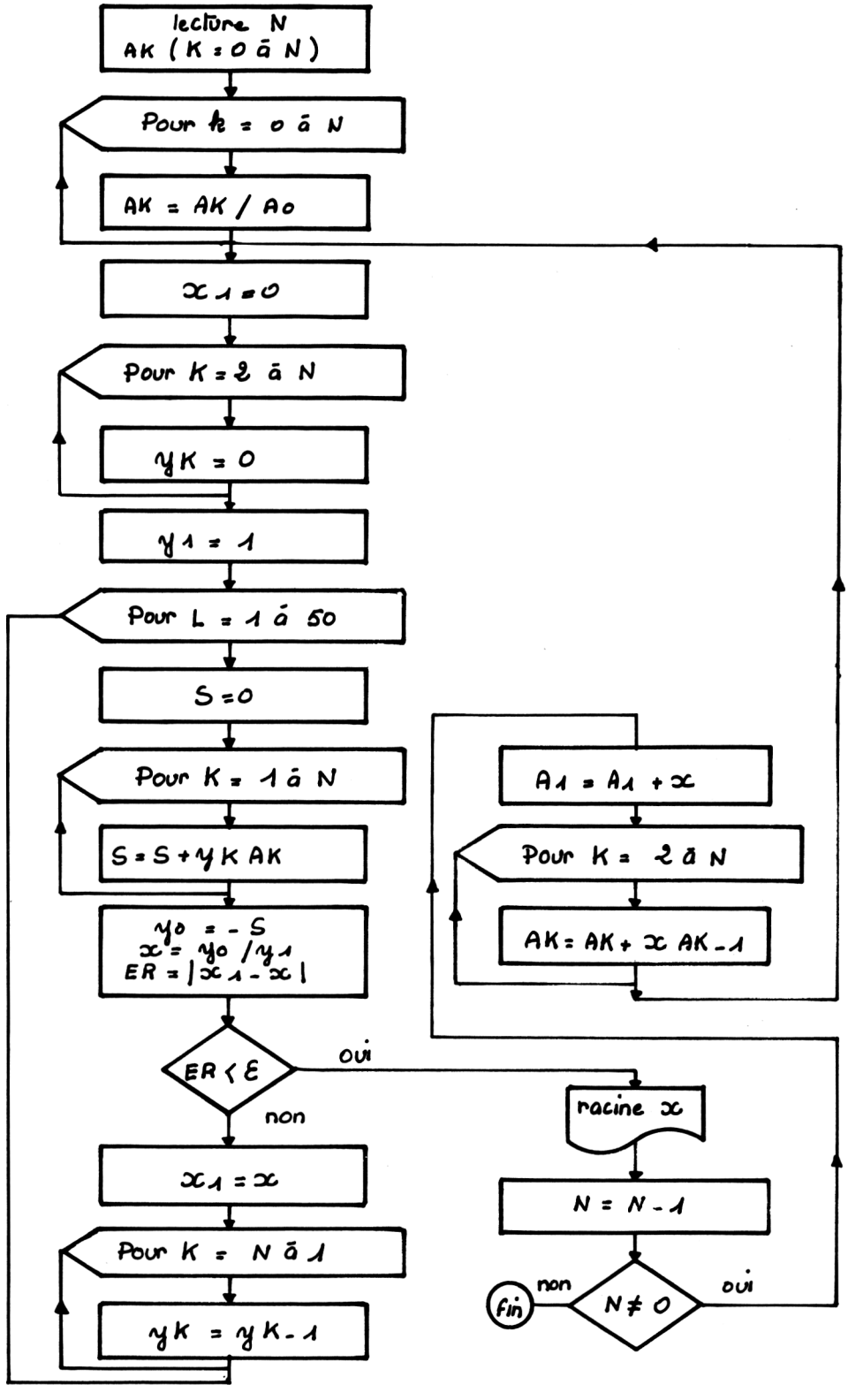
$$\left(\frac{x_2}{x_1}\right)^k , \left(\frac{x_3}{x_1}\right)^k \ldots \longrightarrow 0 \text{ quand } k \to \infty$$

donc : $$\frac{y_{k+1}}{y_k} \longrightarrow x_1 \text{ quand } k \to \infty$$

d'où : $$x_1 = \lim_{k \to \infty} \frac{y_{k+1}}{y_k}$$

Si l'équation aux différences comprend n+1 coefficients, une solution particulière y_k peut être déterminée à partir des n valeurs y_{k-1}, y_{k-2}... y_{n-2}, y_{n-1}, ce qui permet de calculer par récurrence des solutions particulières de l'équation aux différences.

Pour le calcul de la plus grande racine du polynôme, on part des solutions particulières : $y_0=0$, $y_1=0$, $y_2=0$... $y_{n-1}=1$ pour calculer y_n, autre solution particulière. Le même calcul est reconduit pour calculer y_{n+1} à partir de $y_1=0$, $y_2=0$...$y_{n-1}=1$,y_n et ainsi de suite jusqu'à ce que y_{k+1}/y_k ne varie presque plus.


lecture N
AK (K = 0 à N)
Pour k = 0 à N
AK = AK / A0
x1 = 0
Pour K = 2 à N
yK = 0
y1 = 1
Pour L = 1 à 50
S = 0
Pour K = 1 à N
S = S + yK AK
y0 = - S
x = y0 / y1
ER = | x1 - x |
ER < ε
oui
non
x1 = x
Pour K = N à 1
yK = yK - 1
A1 = A1 + x
Pour K = 2 à N
AK = AK + x AK - 1
racine x
N = N - 1
fin
non
N ≠ 0
oui

```
5 REM RACINES D'UN POLYNOME * METHODE DE BERNOUILLI
10 READ ND
20 DIM A(ND),Y(ND)
30 FOR K=0 TO ND:READ A(K):NEXT K
35 N=ND
36 LPRINT "COEFFICIENTS DU POLYNOME"
40 FOR K=0 TO ND:LPRINT "A( ";K;")=";A(K):NEXT K
45 LPRINT "****  R A C I N E S ****"
50 FOR K=0 TO N:A(K)=A(K)/A(0):NEXT K
60 X1=0
80 FOR K=2 TO N:Y(K)=0:NEXT K
90 Y(1)=1
95 FOR L=1 TO 50
100 S=0:FOR K=1 TO N:S=S+Y(K)*A(K):NEXT K
110 Y(0)=-S
120 X=Y(0)/Y(1)
130 ER=ABS(X1-X)
135 IF L<N THEN 150
140 IF ER<5E-5 THEN 190
150 X1=X
160 FOR K=N TO 1 STEP -1:Y(K)=Y(K-1):NEXT K
170 NEXT L
180 PRINT "NON CONVERGENCE  ERREUR=";ER:END
190 LPRINT "X=";X
200 N=N-1:IF N=0 THEN END
210 A(1)=A(1)+X
220 FOR K=2 TO N:A(K)=A(K)+X*A(K-1):NEXT K
230 GOTO 60
3000 DATA 4
3001 DATA 1,2,-13,-14,24
```

```
COEFFICIENTS DU POLYNOME
A(  0 )= 1
A(  1 )= 2
A(  2 )=-13
A(  3 )=-14
A(  4 )= 24
****  R A C I N E S ****
X=-3.99998
X= 2.99997
X=-2.00006
X= 1.00008
```

CALCUL MATRICIEL

1-SYSTEMES D'EQUATIONS LINEAIRES

INTRODUCTION

De nombreux problèmes scientifiques ou techniques se ramènent souvent à la résolution d'un système d'équations linéaires. Un système à trois inconnues par exemple, peut se mettre sous la forme :

$$a_{11}\, x_1 + a_{12}\, x_2 + a_{13}\, x_3 = b_1$$

$$a_{21}\, x_1 + a_{22}\, x_2 + a_{23}\, x_3 = b_2$$

$$a_{31}\, x_1 + a_{32}\, x_2 + a_{33}\, x_3 = b_3$$

Pour ce système d'ordre 3, les trois inconnues sont x_1, x_2, x_3. Les coefficients a sont repérés au moyen de deux indices : i et j. Le premier (i) indique le numéro de ligne, le second (j) le numéro de colonne. Ces coefficients constituent un tableau A appelé matrice associée au système.

Les éléments a_{ij} sont rangés suivant l'ordre donné par les indices :

$$A = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

← a_{23} est le coefficient de la deuxièr ligne, troisième colonne.

Les inconnues x_1, x_2, x_3 et les coefficients b_1, b_2, b_3 sont considérés comme les composantes des vecteurs X et B et s'écrivent sous forme de colonnes :

$$X = \begin{vmatrix} x_1 \\ x_2 \\ x_3 \end{vmatrix} \qquad \text{et } B = \begin{vmatrix} b_1 \\ b_2 \\ b_3 \end{vmatrix}$$

ainsi le système devient :

$$AX = B$$

Cette notation est compacte et générale : cette forme est la même quel que soit le nombre d'inconnues (ou ordre du système). Le produit d'une matrice par un vecteur donne un vecteur. La règle du produit d'une matrice A par un vecteur X est très simple : les composantes i du vecteur B = AX sont obtenues en effectuant les opérations suivantes (pour i de 1 à 3) :

$$b_i = a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3$$

RESOLUTION DES SYSTEMES D'EQUATION PAR ELIMINATION

Il existe de nombreuses méthodes pour résoudre les systèmes d'équations linéaires et l'une des plus classiques, la méthode des déterminants, est à proscrire en calcul numérique, elle conduit à des très grands nombres d'opérations.

Les calculs sont simplifiés lorsque la matrice associée au système d'équations est triangulaire, c'est-à-dire est de la forme :

$$\begin{vmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \qquad \text{ou bien} \qquad \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{vmatrix}$$

Dans le premier cas, elle est dite triangulaire inférieure (souvent notée L - left), dans le second, triangulaire supérieure (R - right).

En écrivant le système AX = B, sous forme développée :

$$\begin{aligned} a_{11}x_1 + 0 \quad x_2 + 0 \quad x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + 0 \quad x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned}$$

x_1 est déterminé immédiatement à partir de la première ligne : $a_{11}x_1=b_1$ d'où $x_1=b_1/a_{11}$ en reportant cette valeur dans la deuxième ligne : $a_{22}x_2 = b_2-a_{21}x_1$.

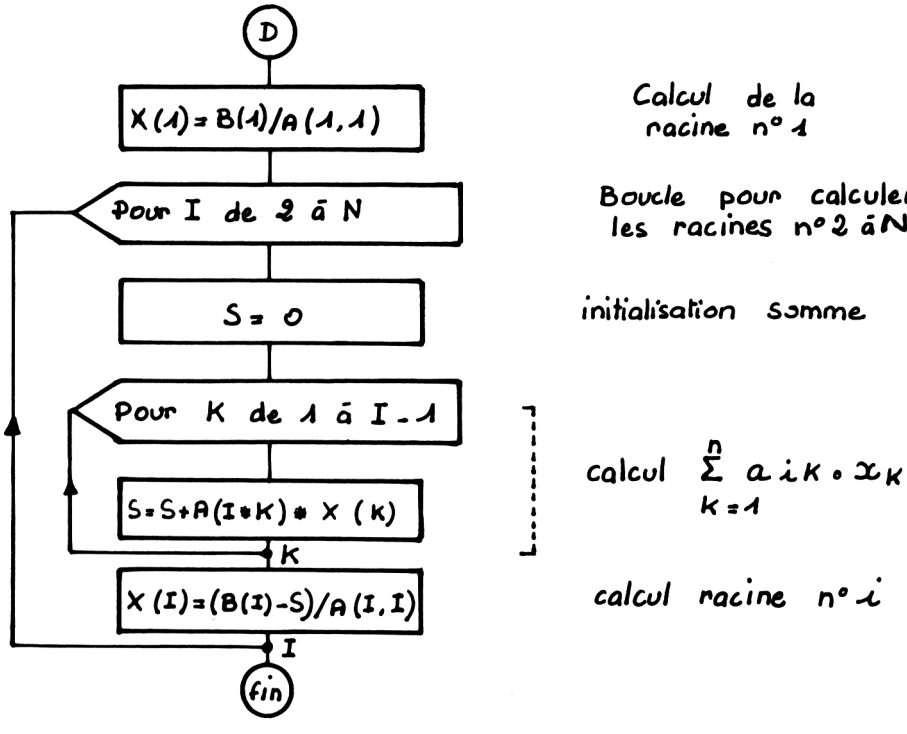
x_2 est calculé sans difficulté : $x_2=(b_2-a_{21}x_1)/a_{22}$ et x_3 est calculé de la même manière :

$$x_3 = (b_3-a_{31}x_1-a_{32}x_2)/a_{33}$$

Le programme BASIC correspondant est très court :

```
 10 DIM A(3,3),X(3),B(3)
    --------------------
    lecture des coefficients aij et bi
    --------------------
100 X(1)=B(1)/A(1,1)
110 X(2)=(B(2)-A(2,1)*X(1))/A(2,2)
120 X(3)=(B(3)-A(3,1)*X1-A(3,2)*X(2))/A(3,3)
```

mais, les systèmes d'équations linéaires peuvent être d'ordre supérieur et le programme doit être plus général.



```
10 REM SYSTEME LINEAIRE TI*X=B
20 REM  TI MATRICE TRIANGULAIRE INFERIEURE
30 DEFINT I-N
40 READ N
50 DIM TI(N,N),X(N),B(N)
60 FOR I=1 TO N :FOR J=1 TO N
70 READ TI(I,J) :NEXT J,I
80 CLS
85 LPRINT "MATRICE TRIANGULAIRE TI"
90 FOR I=1 TO N:LPRINT:FOR J=1 TO N
100 LPRINT TAB((J-1)*7) TI(I,J);:NEXT J,I
110 FOR I=1 TO N:READ B(I):NEXT I
120 LPRINT:LPRINT "SECOND MEMBRE"
130 FOR I=1 TO N:LPRINT "B( ";I;")=";B(I):NEXT I
140 X(1)=B(1)/TI(1,1)
150 FOR I=2 TO N:S=0
160 FOR K=1 TO I-1:S=S+TI(I,K)*X(K):NEXT K
170 X(I)=(B(I)-S)/TI(I,I):NEXT I
175 LPRINT "SOLUTION DE TI*X=B"
180 FOR I=1 TO N:LPRINT "X( ";I;")=";X(I):NEXT I
200 DATA 3
210 DATA 1,0,0,1.5,1,0,3.5,-0.6,1
220 DATA 1,0,0
```

```
MATRICE TRIANGULAIRE TI
 1        0        0
 1.5      1        0
 3.5     -.6       1
SECOND MEMBRE
B(  1 )= 1
B(  2 )= 0
B(  3 )= 0
SOLUTION DE TI*X=B
X(  1 )= 1
X(  2 )=-1.5
X(  3 )=-4.4
```

Le cas d'un système avec matrice triangulaire supérieure TS n'est pas plus compliqué : il suffit de commencer par la dernière ligne et de remonter :

```
110  X(N)=B(N)/TS(N,N)
120  FØR I=N-1 TØ 1 STEP-1
130  S=0
140  FØR K=I+1 TØ N:S=S+TS(I,K)*X(K)
150  NEXT K
160  X(I)=(B(I)-S)/TS(I,I)
170  NEXT I
```

Malheureusement, les systèmes aussi simples sont rares ; les systèmes d'équations linéaires peuvent se mettre sous forme triangulaire si le déterminant est différent de zéro, c'est-à-dire si le système admet une solution.

On sait que toute ligne peut être remplacée par une combinaison linéaire avec d'autres lignes. Ainsi, il est possible par une série de transformations de réduire le système initial à un système à matrice associée triangulaire, c'est le principe de la méthode d'élimination de Gauss.

Reprenons le système :

(1) $a_{11}x_1 + a_{12}\,x_2 + a_{13}\,x_3 = b_1$

(2) $a_{21}x_1 + a_{22}\,x_2 + a_{23}\,x_3 = b_2$

(3) $a_{31}x_1 + a_{32}\,x_2 + a_{33}\,x_3 = b_3$

On multiplie la ligne (1) par $\frac{a_{21}}{a_{11}}$ soit :

$$a_{21}x_1 + \frac{a_{21}}{a_{11}}\,a_{12}x_2 + \frac{a_{21}}{a_{11}}\,a_{13}x_3 = \frac{a_{21}}{a_{11}}\,b_1$$

Le premier terme ($a_{21}x_1$) est devenu égal à celui de la ligne suivante (2). En retranchant la ligne (1) transformée, à la ligne (2), on obtient une nouvelle ligne (2) qui ne comprend plus de terme en x_1 :

$$0x_1 + \left(a_{22} - \frac{a_{21}}{a_{11}}\,a_{12}\right)x_2 + \left(a_{23} - \frac{a_{21}}{a_{11}}\,a_{13}\right)x_3 = b_2 - \frac{a_{21}}{a_{11}}\,b_1$$

On procède de la même manière pour éliminer le terme en x_1 de la troisième ligne : le système devient :

$$\begin{vmatrix} a'_{11} & a'_{12} & a'_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & a'_{32} & a'_{33} \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \end{vmatrix} = \begin{vmatrix} b'_1 \\ b'_2 \\ b'_3 \end{vmatrix}$$

avec $a'_{11}=a_{11}$, $a'_{12}=a'_{13}$, $a'_{13}=a_{13}$, $b'_1=b_1$

$$a'_{22}=a_{22}-\frac{a_{21}}{a_{11}}a_{12} \quad , \quad a'_{23}=a_{23}-\frac{a_{21}}{a_{11}}a_{13}, \quad b'_2=b_2-\frac{a_{21}}{a_{11}}b_1$$

$$a'_{32}=a_{32}-\frac{a_{31}}{a_{11}}a_{12} \quad , \quad a'_{33}=a_{33}-\frac{a_{31}}{a_{11}}a_{13}, \quad b'_3=b_3-\frac{a_{31}}{a_{11}}b_1$$

On élimine le terme en x_2 de la troisième ligne en procédant encore de la même manière. On multiplie la deuxième ligne du système transformé A'X = B' par $\frac{a'_{32}}{a'_{22}}$ et on soustrait cette ligne à la troisième ligne, le terme en x_2 disparaît et la matrice devient triangulaire supérieure :

$$(A"X) \quad \begin{vmatrix} a"_{11} & a"_{12} & a"_{13} \\ 0 & a"_{22} & a"_{23} \\ 0 & 0 & a"_{33} \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \end{vmatrix} = \begin{vmatrix} b"_1 \\ b"_2 \\ b"_3 \end{vmatrix} \quad (B")$$

avec $a"_{11}=a'_{11}$, $a"_{12}=a'_{12}$, $a"_{13}=a'_{13}$, $b"_1=b'_1$

$a"_{22}=a'_{22}$, $a"_{23}=a'_{23}$, $b"_2=b'_2$

$$a"_{33}=a'_{33}-\frac{a'_{32}}{a'_{22}}a'_{23} \quad , \quad b"_3=b'_3-\frac{a'_{33}}{a'_{22}}b'_2$$

Ces opérations ne sont réalisables que si :

$$a_{11} \neq 0 \quad , \quad a'_{22} \neq 0$$

ces coefficients particuliers sont appelés pivots.

Pour être sûr de ne pas avoir de division par zéro, il faut tester le pivot et éventuellement permuter les lignes. Cet algorithme détruit la matrice d'origine A.

```
10 REM  SYSTEME LINEAIRE  A*X=B
20 DEFINT I-N
30 READ N
40 DIM A(N,N),X(N),B(N)
50 FOR I=1 TO N: FOR J=1 TO N
60 READ A(I,J):NEXT J,I
70 CLS
75 LPRINT "MATRICE A"
80 FOR I=1 TO N:LPRINT:FOR J=1 TO N
90 LPRINT TAB((J-1)*7) A(I,J);:NEXT J,I
100 FOR I=1 TO N:READ B(I):NEXT I
110 LPRINT:LPRINT "SECOND MEMBRE"
130 FOR I=1 TO N:LPRINT "B( ";I;")=";B(I):NEXT I
140 REM TRANSFORMATION EN MATRICE TRIANGULAIRE
150 FOR K=1 TO N-1
160 FOR I=K+1 TO N
170 B(I)=B(I)-A(I,K)/A(K,K)*B(K)
180 FOR J=K+1 TO N
190 A(I,J)=A(I,J)-A(I,K)/A(K,K)*A(K,J)
200 NEXT J,I
210 NEXT K
220 REM RESOLUTION DU SYSTEME TRIANGULAIRE
230 X(N)=B(N)/A(N,N)
240 FOR I=N-1 TO 1 STEP -1
250 S=0
260 FOR K=I+1 TO N:S=S+A(I,K)*X(K)
270 NEXT K
280 X(I)=(B(I)-S)/A(I,I):NEXT I
285 LPRINT "SOLUTION DE AX=B"
290 FOR I=1 TO N:LPRINT "X( ";I;")=";X(I)
300 NEXT I
310 END
320 DATA 3
330 DATA 2,1,-2,3,-1,1,7,5,-3
340 DATA 1,0,0
```

```
MATRICE A
 2        1       -2
 3       -1        1
 7        5       -3
SECOND MEMBRE
B(  1 )= 1
B(  2 )= 0
B(  3 )= 0
SOLUTION DE AX=B
X(  1 )= .0625
X(  2 )=-.5
X(  3 )=-.6875
```

METHODE DE CROUT - DECOMPOSITION

Lorsqu'on est amené à utiliser la même matrice pour résoudre plusieurs fois le système, il est intéressant d'employer une méthode assez voisine qui consiste à factoriser la matrice A : la matrice A associée au système, est remplacée par le produit de deux matrices notées L-R (Left - Right).

Ces matrices sont respectivement triangulaire inférieure et triangulaire supérieure. Les coefficients de la diagonale de L(lkk) sont égaux à 1.

$$A = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \quad L = \begin{vmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{vmatrix} \quad R = \begin{vmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{vmatrix}$$

Rappelons les règles de calcul du produit de deux matrices A.B :

$$A = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \quad B = \begin{vmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{vmatrix} \quad C=A.B = \begin{vmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{vmatrix}$$

avec : $c_{11} = a_{11}.b_{11}+a_{12}.b_{21}+a_{13}.b_{31}$

$c_{12} = a_{11}.b_{12}+a_{12}.b_{22}+a_{13}.b_{32}$

$c_{13} = a_{11}.b_{13}+a_{12}.b_{23}+a_{13}.b_{33}$

$c_{21} = a_{21}.b_{11}+a_{22}.b_{21}+a_{23}.b_{31}$

---------- etc.

D'une manière plus générale : $A(a_{ij}), B(b_{ij}), C(C_{ij})$

$$C_{ij} = \sum_{k=1}^{n} a_{ik}.b_{kj}$$

L'inverse A^{-1} de la matrice A est une matrice telle que : $A.A^{-1} = I$.

où I est la matrice unité : tous les coefficients de la diagonale sont égaux à 1, les autres sont nuls :

$$I = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Appliquons cette règle au produit LR et identifions les coefficients du produit obtenu à ceux de A.

$$\begin{vmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{vmatrix} \quad \begin{vmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{vmatrix} = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

$$L \qquad\qquad R \qquad = \qquad A$$

$a_{11}=1.r_{11}+0.0+0.0$

$=r_{11}$

$a_{12}=r_{12}$, $a_{13} = r_{13}$

$a_{21}=l_{21}r_{11}$, $a_{31} = l_{31}r_{11}$

$a_{22}=l_{21}.r_{12}+r_{22}$, $a_{23} = l_{21}.r_{13}+r_{23}$

$a_{32}=l_{31}.r_{12}+l_{32}.r_{22}$ et $a_{33}=l_{31}r_{13}+l_{32}.r_{23}+r_{33}$

Les coefficients indéterminés l_{ij} et r_{ij} peuvent tous être calculés à partir de a_{ij} à condition de respecter rigoureusement l'ordre de calcul suivant :

calcul de r_{11} , r_{12} , r_{13}

puis de l_{21} , l_{31}

et de r_{22} , r_{23}

l_{32}

r_{33}

et d'une manière générale :

$r_{1i}=a_{1i}$ pour i de 1 à n

$l_{i1}=a_{i1}/a_{11}$ pour i de 1 à n

pour k de 2 à n.

pour j de k à n : $r_{kj}=a_{kj} - \sum_{i=1}^{k-1} l_{ki}.r_{ij}$

pour i de k+1 à n (tant que $k < n$)

$$l_{ik}= \left(a_{ik} - \sum_{j=1}^{k-1} l_{ij}.r_{jk}\right)/r_{kk}$$

ce qui se traduit par l'organigramme ci-après.

Une vérification peut être faite en effectuant le produit P=L.R dont les coefficients doivent s'identifier à ceux de A.

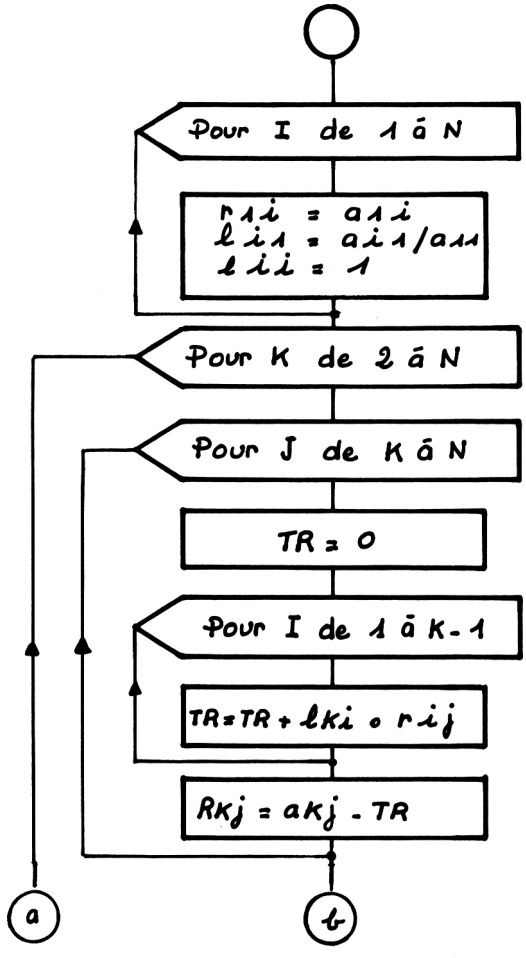
Le système AX = B devient donc : LRX = B

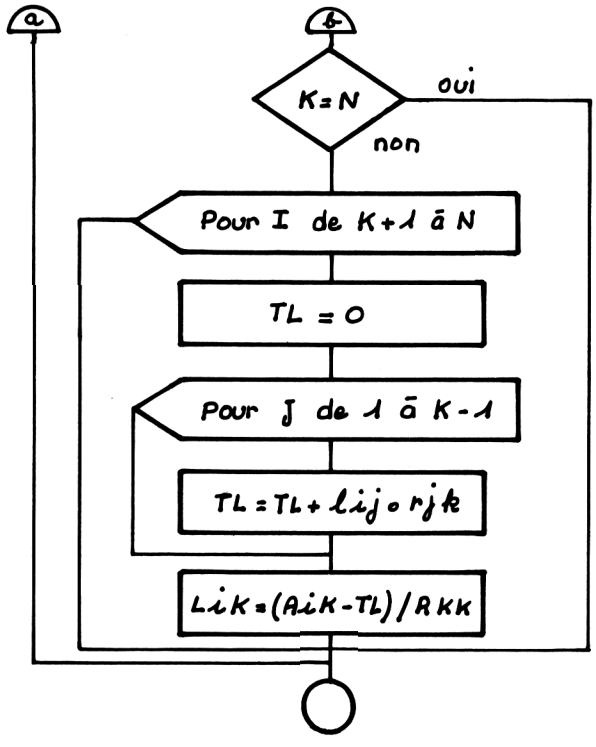
En posant RX = Y, le système peut être résolu en deux étapes :

- calcul de Y (à partir de L et B) : LY = B
- calcul de X (à partir de R et Y) : RX = Y

C'est la méthode de Crout.

Exercice : *écrire le programme correspondant à la méthode de Crout ; pour cela, il suffit de compléter le programme de décomposition ci-après.*






```
10 REM FACTORISATION L*R=A
20 DEFINT I-K,N
30 READ N
40 DIM A(N,N),L(N,N),R(N,N)
50 FOR I=1 TO N:FOR J=1 TO N
60 READ A(I,J):NEXT J,I
70 CLS
75 LPRINT "MATRICE A";
80 FOR I=1 TO N:LPRINT:FOR J=1 TO N
90 LPRINT TAB((J-1)*7) A(I,J);:NEXT J,I
100 FOR I=1 TO N:L(I,I)=1:NEXT I
110 FOR J=1 TO N:R(1,J)=A(1,J):NEXT J
120 FOR I=2 TO N:L(I,1)=A(I,1)/A(1,1):NEXT I
130 FOR K=2 TO N:FOR J=K TO N:TR=0
140 FOR I=1 TO K-1
145 TR=TR+R(I,J)*L(K,I):NEXT I
150 R(K,J)=A(K,J)-TR:NEXT J
160 IF K=N THEN 220
170 FOR I=K+1 TO N:TL=0
180 FOR J=1 TO K-1
190 TL=TL+L(I,J)*R(J,K):NEXT J
200 L(I,K)=(A(I,K)-TL)/R(K,K)
210 NEXT I
220 NEXT K
225 LPRINT:LPRINT"MATRICE L";
230 FOR I=1 TO N:LPRINT:FOR J=1 TO N
240 LPRINT TAB((J-1)*7) L(I,J);:NEXT J,I
```

```
245 LPRINT:LPRINT "MATRICE R";
250 FOR I=1 TO N:LPRINT:FOR J=1 TO N
260 LPRINT TAB((J-1)*7) R(I,J);:NEXT J,I
270 REM VERIFICATION P=L*R
280 DIM P(N,N)
290 FOR I=1 TO N:FOR J=1 TO N:S=0
300 FOR K=1 TO N:S=S+L(I,K)*R(K,J):NEXT K
310 P(I,J)=S:NEXT J,I
315 LPRINT:LPRINT "MATRICE P=L*R";
320 FOR I=1 TO N:LPRINT:FOR J=1 TO N
330 LPRINT TAB((J-1)*7) P(I,J);:NEXT J,I
340 END
350 DATA 3
360 DATA 2,1,-2,3,-1,1,7,5,-3
```

```
MATRICE A
 2       1      -2
 3      -1       1
 7       5      -3
MATRICE L
 1       0       0
 1.5     1       0
 3.5    -.6      1
MATRICE R
 2       1      -2
 0      -2.5     4
 0       0       6.4
MATRICE P=L*R
 2       1      -2
 3      -1       1
 7       5      -3
```

INVERSION DE MATRICES

L'inverse A^{-1} d'une matrice A d'ordre n peut être calculé en résolvant n fois un système d'équations. Soit Ii un vecteur où toutes les composantes sont nulles sauf la ième qui vaut 1 :

$$I_i = \begin{vmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{vmatrix} \longleftarrow \text{composante i}$$

En multipliant à gauche par A^{-1}, le système $AX = I_i$ devient :

$$X = A^{-1}I_i$$

(puisque $A^{-1}A=I$ et $IX = X$)

en développant :

$$\begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_i \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{vmatrix} = \begin{vmatrix} a_{11}^{-1} & a_{12}^{-1} & \cdots & a_{1i}^{-1} & \cdots & a_{1n}^{-1} \\ a_{21}^{-1} & a_{22}^{-1} & \cdots & a_{2i}^{-1} & \cdots & a_{2n}^{-1} \\ a_{31}^{-1} & a_{32}^{-1} & \cdots & a_{3i}^{-1} & \cdots & a_{3n}^{-1} \\ \cdot & \cdot & & & & \\ \cdot & \cdot & & & & \\ \cdot & \cdot & & & & \\ a_{i1}^{-1} & a_{i2}^{-1} & \cdots & a_{ii}^{-1} & \cdots & a_{in}^{-1} \\ \cdot & \cdot & & & & \\ \cdot & \cdot & & & & \\ \cdot & \cdot & & & & \\ a_{n1}^{-1} & a_{n2}^{-1} & \cdots & a_{ni}^{-1} & \cdots & a_{nn}^{-1} \end{vmatrix} \cdot \begin{vmatrix} 0 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{vmatrix} = \begin{vmatrix} a_{1i}^{-1} \\ a_{2i}^{-1} \\ a_{3i}^{-1} \\ \cdot \\ \cdot \\ \cdot \\ a_{ii}^{-1} \\ \cdot \\ \cdot \\ \cdot \\ a_{ni}^{-1} \end{vmatrix}$$

Donc, la solution d'un système, en prenant comme second membre I_i donne la ième colonne de la matrice inverse.

En conséquence, pour calculer l'inverse de la matrice A, il suffit de résoudre le système en prenant comme second membre $I_1, I_2 \ldots I_n$, donc n systèmes à résoudre.

Une autre méthode est aussi fréquemment utilisée : si une matrice A non singulière (qui admet une solution pour AX=B) peut être réduite à une matrice identité par une suite de transformations de la forme $E_k.E_{k-1}\ldots E_1$, alors en appliquant ces mêmes transformations à I, matrice identité, on obtient A^{-1}, matrice inverse de A; la démonstration est très simple :

si $E_k.E_{k-1} \ldots E_1.A = I$

en multipliant à droite par A^{-1}, on obtient :

$$E_k.E_{k-1}. \ldots E_1.A.A^{-1} = IA^{-1}$$

or $AA^{-1} = I$ et $IA^{-1} = A^{-1}$

donc $E_k.E_{k-1}. \ldots E_1.I = A^{-1}$

Etudions trois matrices particulières E :

1 - $E_i(c)$: cette matrice est presque de la même forme qu'une matrice identité (où les coefficients de la diagonale sont égaux à 1 et les autres sont nuls), mais la ligne i a un coefficient égal à c au lieu de 1.

$$E_i(c) = \begin{vmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & 0 & \\ & & & \ddots & & \\ & & & & c & \text{---- ligne i} \\ & & 0 & & \ddots & \\ & & & & & 1 \\ & & & & & & 1 \end{vmatrix}$$

Le produit d'une matrice A par $E_i(c)$ donne une matrice A' :

$$A' = E_i(c).A$$

En considérant les lignes de A, notées $R_1, R_2, R_3 \ldots R_n$ et les lignes de A' notées $R'_1, R'_2, R'_3 \ldots R_n$, cette matrice $E_i(c)$ transforme ces lignes suivant la règle :

$R'_i = c\ R_i$ pour la ligne i

$R'_i = R_i$ pour les autres lignes

ou encore, si a_{ij} sont les coefficients de A et a'_{ij} sont les coefficients de A' :

$a'_{ij} = ca_{ij}$ pour la ligne i, j de 1 à n

$a'_{kj} = a_{kj}$ pour les autres lignes, j de 1 à n ($k \neq i$)

2 - E_{ik} : cette matrice provient d'une matrice identité où les lignes i et k ont été permutées :

$$E_{ik} = \begin{vmatrix} 1 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & \text{ligne i} \\ 0 & 1 & 0 & 0 & \\ 0 & 0 & 0 & 1 & \text{ligne k} \end{vmatrix}$$

Si $A' = E_{ik}$, alors $R'_i = R_k$

$R'_k = R_i$

et $R'_m = R_m$ pour $m \neq i,k$

ou encore :

$a'_{ij} = a_{kj}$, $a'_{kj} = a_{ij}$ pour j de 1 à n

$a'_{mj} = a_{mj}$ pour les autres lignes $m \neq k,i$, pour j de 1 à n

3 - $E_{ik}(c)$: cette matrice provient d'une matrice identité où la ligne i est remplacée par la somme de la ligne i et de la ligne k multipliée par c (constante).

si $A' = E_{ik}(c).\ A$, alors $R'_i = R_i + c\ R_k$

et $R'_m = R_m$ pour $m \neq i$

ou encore :

$a'_{ij} = a_{ij} + c\ a_{kj}$ pour j de 1 à n

$a'_{mj} = a_{mj}$ pour m de 1 à n sauf m=i et pour j de 1 à n

La méthode d'élimination pour obtenir la matrice inverse A^{-1} est effectuée en plusieurs étapes (n) comprenant chacune deux phases : pour l'étape k :
a- normalisation de l'élément $a_{k,k}$ en multipliant la ligne par son inverse (on applique à la matrice $A^{(k)}$ la transformation $E_k(1/a_{kk})$
b- mise à zéro des éléments sur et sous la diagonale jusqu'à la colonne k : application à la matrice $E_{ik}(-a_{ik})$

Cet algorithme n'est donc pas compliqué, pour s'en rendre compte, il suffit de suivre pas à pas les transformations. On part de A :

$$A = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & & & \\ \vdots & & & \\ \vdots & & & \\ a_{n-1} & & & a_{nn} \end{vmatrix}$$

1ère étape

phase a : $R'_1 = R_1/a_{11}$

$$\text{d'où } A^{(1)} = \begin{vmatrix} 1 & a'_{12} & a'_{13} & \dots & a'_{1n} \\ a'_{21} & & & & \\ \vdots & & & & \\ a'_{n1} & \dots & \dots & \dots & a'_{nn} \end{vmatrix}$$

avec $a'_{1j} = a_{1j}/a_{11}$ pour la première ligne
et $a'_{ij} = a_{ij}$ pour les autres

. *phase b :* $R_i^{(1)} = R_i - a_{i1}\, R_i^{(1)}$ pour $i \neq 1$

$$\text{d'où } A^{(1)} = \begin{vmatrix} 1 & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ \vdots & & & \\ 0 & \dots & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{vmatrix}$$

avec $a_{ij}^{(1)} = a_{ij} - a_{i1}.a_{ij}$ pour $i \neq 1$

Les calculs de la **2ème étape** conduisent à :

$$A^{(2)} = \begin{vmatrix} 1 & 0 & a_{13}^{(2)} & \dots & a_{1n}^{(2)} \\ 0 & 1 & a_{23}^{(2)} & \dots & \\ \vdots & \vdots & & & \\ & 0 & & & \\ \vdots & \vdots & & & \\ 0 & 0 & a_{n3}^{(2)} & \dots & a_{nn}^{(2)} \end{vmatrix}$$

Pour **l'étape k,** la matrice $A^{(k-1)}$ est transformée de la même manière :

. pour la ligne k : $a_{kj}^{(k)} = a_{kj}^{(k-1)} / a_{kk}^{(k-1)}$ pour j de 1 à
évidemment la division ne peut se faire que si :
$a_{kk}^{(k-1)} \neq 0$ ($a_{kk}^{(k-1)}$ est le pivot)

. pour les autres lignes : i de 1 à n sauf i=k

$a_{ij}^{(k)} = a_{ij}^{(k-1)} - a_{ik}^{(k-1)} \cdot a_{kj}^{(k)}$ pour j de 1 à n

Exercice : *montrer que* $a_{ik}^{(k)} = 0$ *quel que soit i différent de k*

et pour la matrice $B^{(k-1)}$

. pour la ligne k : $b_{kj}^{(k)} = b_{kj}^{(k-1)} / a_{kk}^{(k-1)}$ j de 1 à n

. pour les autres lignes : i de 1 à n sauf i=k

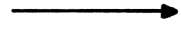
$b_{ij}^{(k)} = b_{ij}^{(k-1)} - a_{ik}^{(k-1)} \cdot b_{kj}^{(k)}$ pour j de 1 à n

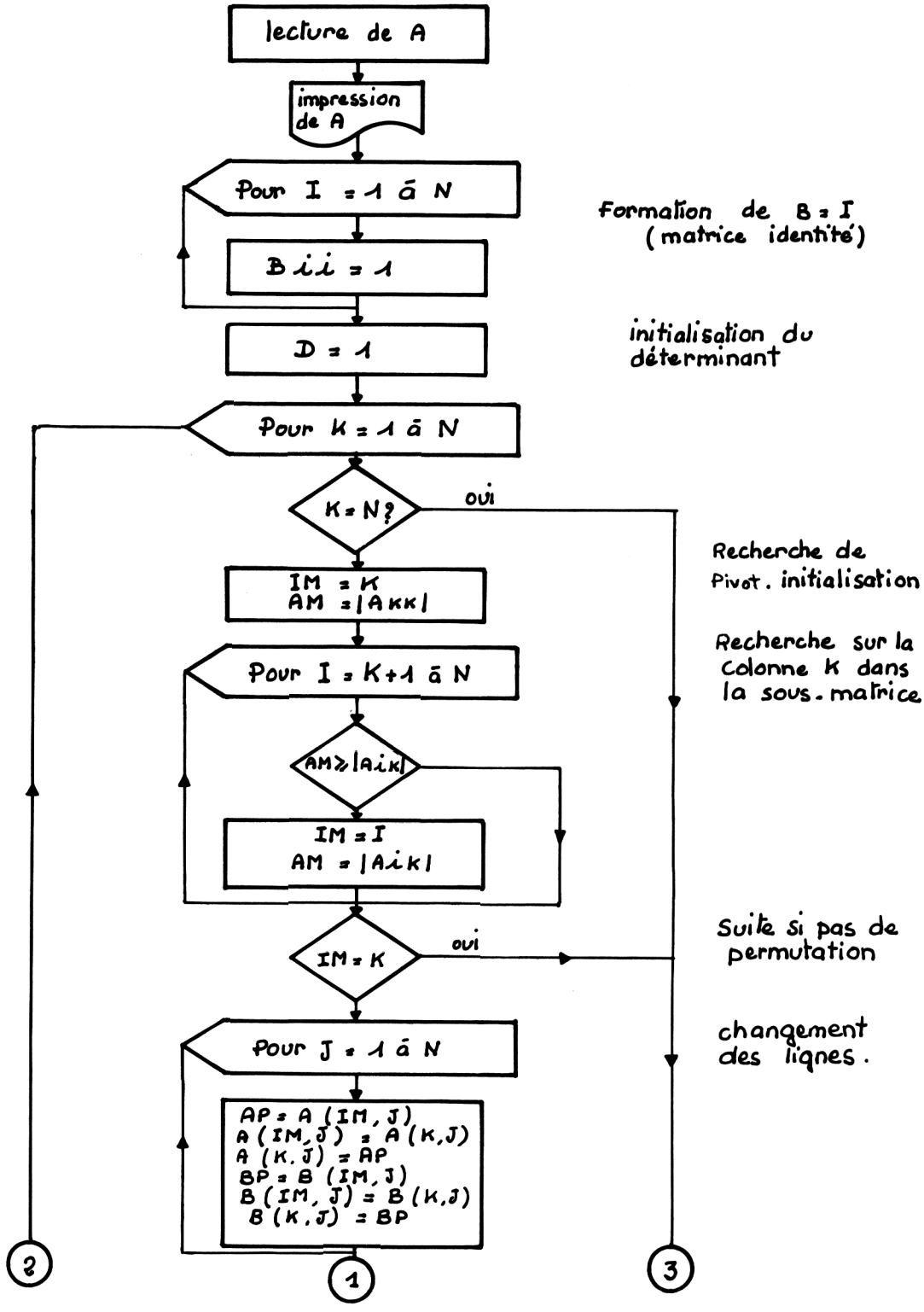
lorsque k = n , $A^{(k)}$ est devenu matrice identité I et $B^{(k)}$, la matrice A^{-1} recherchée.

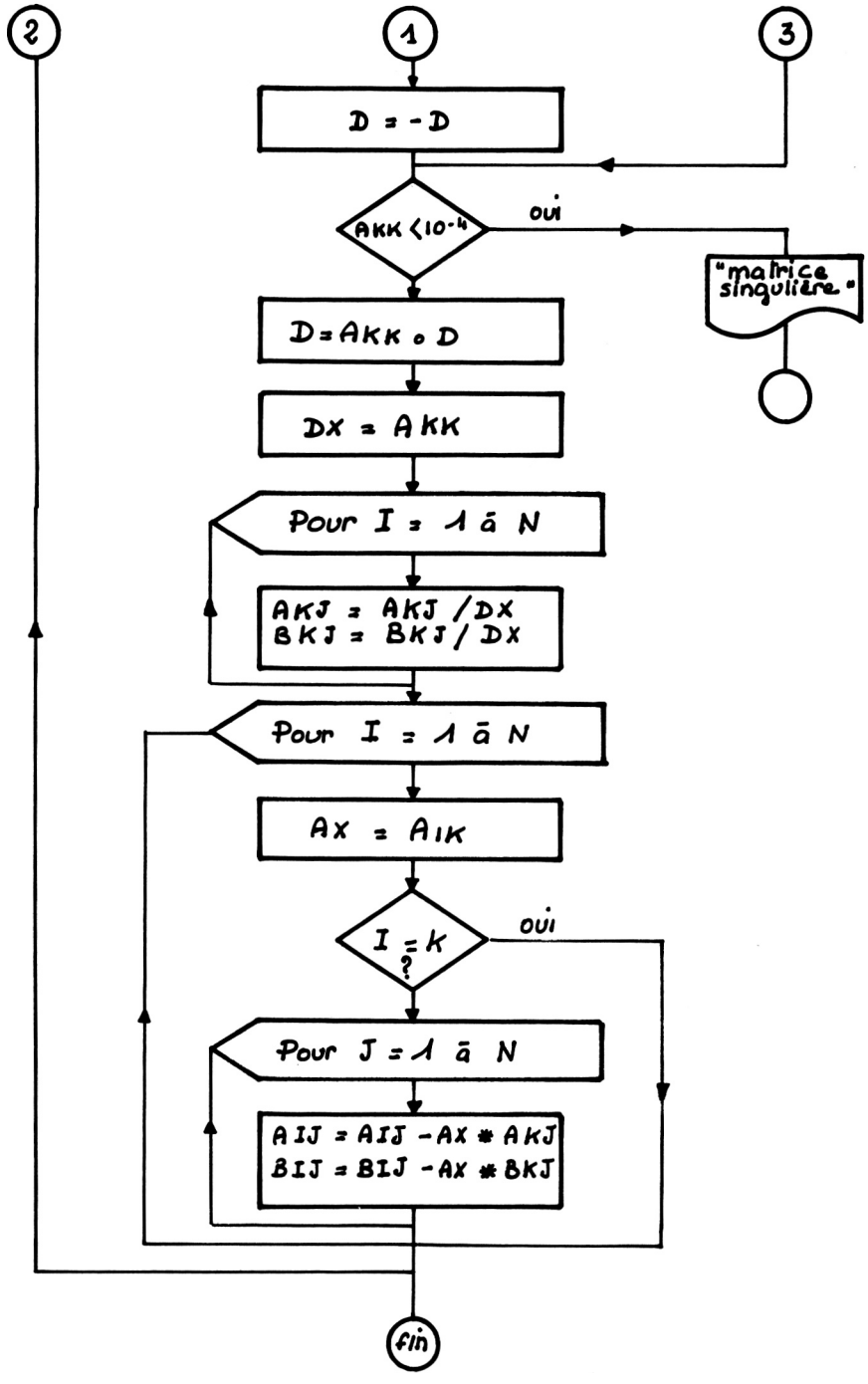
Remarques :

1 - L'organigramme ci-après, ainsi que le programme BASIC, correspondent à l'algorithme tel qu'il vient d'être présenté. Mais, un examen attentif du déroulement des calculs montre qu'il est possible de réduire le nombre de calculs (gain de temps) et l'encombrement mémoire en n'utilisant qu'une seule matrice : la matrice A qui se transforme pas à pas en A^{-1}.

2 - Ce programme permet aussi le calcul du déterminant qui correspond au produit des pivots.




lecture de A
impression de A
Pour I = 1 à N
Bii = 1
D = 1
Pour K = 1 à N
K = N ?
oui
IM = K
AM = |AKK|
Pour I = K+1 à N
AM ≥ |AiK|
IM = I
AM = |AiK|
IM = K
oui
Pour J = 1 à N
AP = A (IM, J)
A (IM, J) = A (K, J)
A (K, J) = AP
BP = B (IM, J)
B (IM, J) = B (K, J)
B (K, J) = BP
2
1
3
Formation de B = I (matrice identité)
initialisation du déterminant
Recherche de Pivot. initialisation
Recherche sur la colonne K dans la sous-matrice
Suite si pas de permutation
changement des lignes.


2
1
3
D = -D
AKK <10-4
oui
"matrice singulière"
D=AKK o D
DX = AKK
Pour I = 1 à N
AKJ = AKJ /DX
BKJ = BKJ / DX
Pour I = 1 à N
AX = AIK
I = K ?
oui
Pour J = 1 à N
AIJ = AIJ -AX * AKJ
BIJ = BIJ -AX * BKJ
fin

```
10 REM INVERSION DE MATRICE
20 DEFINT I-N
30 READ N
40 DIM A(N,N),B(N,N)
50 FOR I=1 TO N:FOR J=1 TO N
60 READ A(I,J):NEXT J,I
70 CLS
80 LPRINT "MATRICE A"
90 FOR I=1 TO N:LPRINT:FOR J=1 TO N
100 LPRINT TAB((J-1)*7) A(I,J);:NEXT J,I
110 REM MATRICE B UNITAIRE
120 FOR I=1 TO N:B(I,I)=1:NEXT I
125 D=1
130 FOR K=1 TO N
140 IFK=N THEN 310
150 IM=K
160 AM=ABS(A(K,K))
170 FOR I=K+1 TO N
180 IF AM)=ABS(A(K,K)) THEN 210
190 IM=I
200 AM=ABS(A(I,K))
210 NEXT I
215 REM ECHANGE COL K ET IM SI IM()K
220 IF IM=K THEN 300
230 FOR J=1 TO N
240 AP=A(IM,J)
250 A(IM,J)=A(K,J):A(K,J)=AP
260 BP=B(IM,J)
270 B(IM,J)=B(K,J)
280 B(K,J)=BP
290 NEXT J:D=-D
300 PRINT
310 IF ABS(A(K,K))(1.E-6 PRINT"MAT SING";K;A(K,K)
320 D=A(K,K)*D
330 DX=A(K,K)
340 FOR J=1 TO N
350 A(K,J)=A(K,J)/DX
360 B(K,J)=B(K,J)/DX
370 NEXT J
380 REM ELIMINATION PAR COMBINAISON LINEAIRE
390 FOR I=1 TO N
400 AX=A(I,K)
410 IF I=K THEN 460
420 FOR J=1 TO N
430 A(I,J)=A(I,J)-AX*A(K,J)
440 B(I,J)=B(I,J)-AX*B(K,J)
450 NEXT J
460 NEXT I
470 NEXT K
```

```
480 LPRINT:LPRINT"MATRICE INVERSE";
490 FOR I=1 TO N:LPRINT:FOR J=1 TO N
500 LPRINT TAB((J-1)*7) B(I,J);:NEXT J,I
504 LPRINT:LPRINT
505 LPRINT"DETERMINANT  ";D
506 LPRINT
510 DIM P(N,N)
520 REM RESTAURATION A
530 RESTORE
540 READ N
550 FOR I=1 TO N:FOR J=1 TO N
560 READ A(I,J):NEXT J,I
570 FOR I=1 TO N:FOR J=1 TO N
580 S=0:FOR K=1 TO N:S=S+A(I,K)*B(K,J):NEXT K
590 P(I,J)=S:NEXT J,I
600 LPRINT:LPRINT"VERIFICATION P=A*B";
610 FOR I=1 TO N:LPRINT :FOR J=1 TO N
620 LPRINT TAB((J-1)*7) P(I,J);:NEXT J,I
630 END
640 DATA 3
650 DATA 9,9,8,9,8,7,8,7,6
```

```
MATRICE A
 9        9        8
 9        8        7
 8        7        6
MATRICE INVERSE
-1        2       -1
 2      -10        9
-1        9       -9
DETERMINANT    1
VERIFICATION P=A*B
 1       -7.62939E-06   7.62939E-06
 9.53674E-07   1   0
 9.53674E-07 -3.8147E-06   1
```

CALCUL MATRICIEL
2- VALEURS ET VECTEURS PROPRES METHODES ITERATIVES POUR SYSTEMES LINEAIRES

VALEURS PROPRES ET VECTEURS PROPRES

Pour résoudre certains problèmes de physique (vibrations ...), pour étudier des méthodes de calcul itératives, il faut rechercher les valeurs propres et vecteurs propres d'une matrice.

Le scalaire λ est valeur propre et V le vecteur propre associé si :

$$AV = \lambda V$$

Les valeurs propres sont les racines du polynôme $P_n(\lambda) = \text{dét}(A - \lambda I)$ appelé polynôme caractéristique, de degré n, égal à l'ordre de la matrice. Les racines peuvent être simples ou multiples ; si elles sont distinctes, la matrice peut être réduite à la forme diagonale D et il existe une matrice T telle que :

$$D = T^{-1}AT$$

Les algorithmes de calcul des valeurs propres et vecteurs propres sont très nombreux.

METHODE DE LA "PUISSANCE ITEREE"

Cet algorithme permet de calculer le module de la plus grande valeur propre et le vecteur propre associé.

Si la matrice est définie positive, c'est-à-dire si ses valeurs propres sont distinctes et positives, le calcul donnera la plus grande valeur propre λ_1 :

$$\lambda_1 > \lambda_2 > \lambda_3 \ldots \lambda_n$$

Les vecteurs propres constituent une base et un vecteu quelconque X_0 peut s'écrire :

$$X_0 = c_1 V_1 + c_2 V_2 + c_3 V_3 \ldots + c_n V_n$$

où $V_1, V_2, V_3 \ldots V_n$ sont les vecteurs propres correspondant à $\lambda_1, \lambda_2, \lambda_3 \ldots \lambda_n$ et $c_1, c_2, c_3 \ldots c_n$ les composantes de X_0.

En multipliant A par X_0, on obtient :

$$AX_0 = Ac_1 V_1 + Ac_2 V_2 + Ac_3 V_3 + \ldots + Ac_n V_n$$

or :

$$Ac_1 V_1 = c_1 AV_1 = c_1 \lambda_1 V_1 \ , \ Ac_2 V_2 = \lambda_2 V_2 \ \ldots$$

donc :

$$AX_0 = c_1 \lambda_1 V_1 + c_2 \lambda_2 V_2 + \ldots + c_n \lambda_n V_n$$

En multipliant de nouveau par A :

$$A^2X_0 = c_1 \lambda_1 AV_1 + c_2 \lambda_2 AV_2 + c_3 \lambda_3 AV_3 + \ldots + c_n \lambda_n AV_n$$
$$= c_1 \lambda_1^2 V_1 + c_2 \lambda_2^2 V_2 + c_3 \lambda_3^2 V_3 + \ldots + c_n \lambda_n^2 V_n$$

et ainsi de suite, d'où :

$$A^kX_0 = c_1 \lambda_1^k V_1 + c_2 \lambda_2^k V_2 + c_3 \lambda_3^k V_3 + \ldots + c_n \lambda_n^k V_n$$

et en mettant λ_1^k en facteur :

$$A^kX_0 = \lambda_1^k \left[c_1 V_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k V_2 + c_3 \left(\frac{\lambda_3}{\lambda_1}\right)^k V_3 + \ldots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k V_n \right]$$

quand k devient très grand, comme $\lambda_2 < \lambda_1$, $\lambda_3 < \lambda_1$, $\ldots \lambda_n < \lambda_1$, $\left(\frac{\lambda_2}{\lambda_1}\right)^k$ tend vers 0 :

$$\left(\frac{\lambda_2}{\lambda_1}\right)^k \rightarrow 0 \text{ quand } k \rightarrow \infty$$

d'où : $A^kX_0 \rightarrow \lambda_1^k c_1 V_1$

et : $A^{k+1}X_0 \rightarrow \lambda_1^{k+1} c_1 V_1$

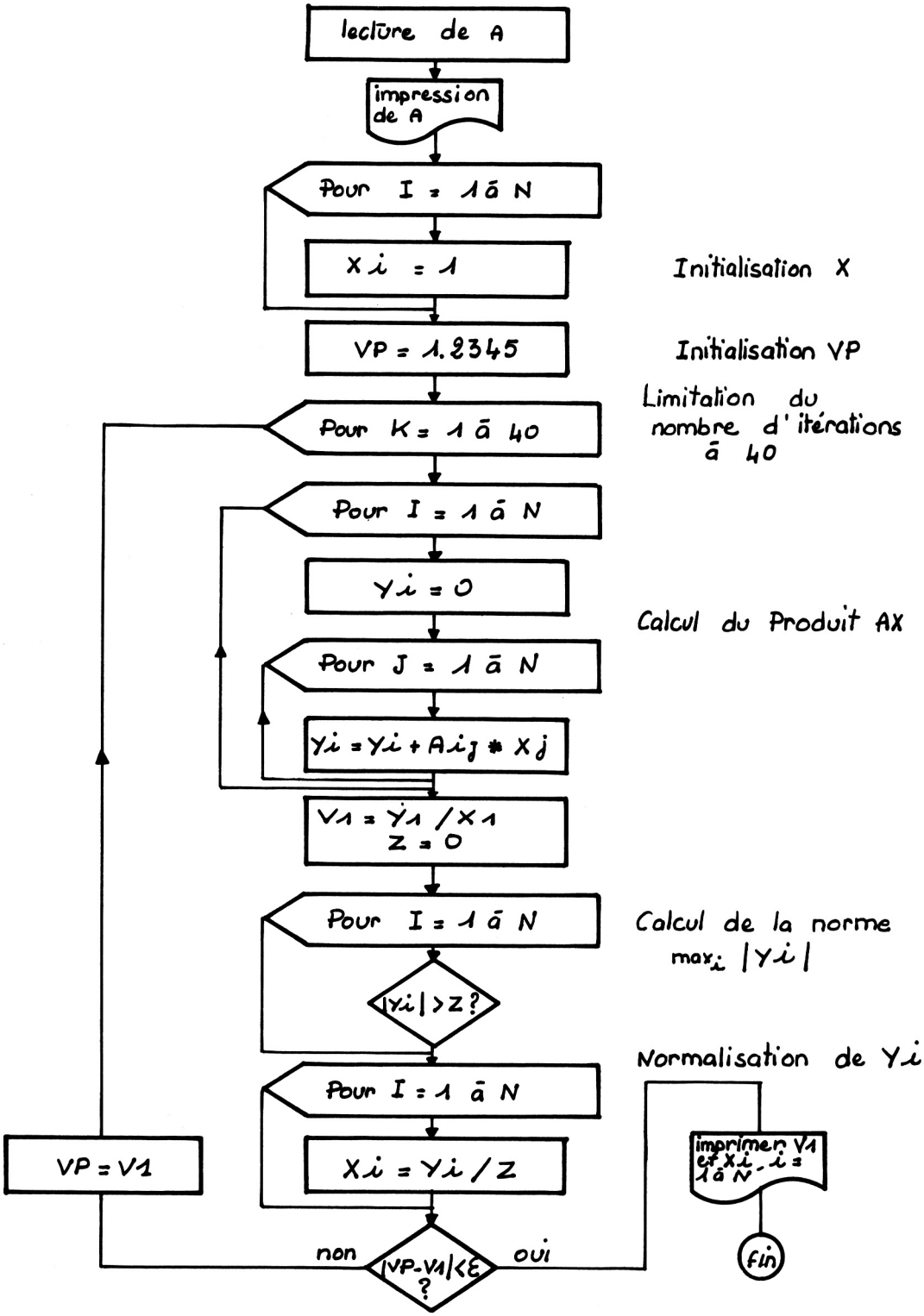
donc pour toutes les composantes i des vecteurs A^kX_0 et $A^{k+1}X_0$:

$$\frac{(A^{k+1}X_0)_i}{(A^kX_0)_i} \rightarrow \lambda_1$$

C'est le principe de calcul de λ_1. Si $Y_k=A^kX_0$, $Y_k \rightarrow \lambda_1^k c_1 V_1$, cela signifie que y_k tend vers le vecteur propre associé à V_1.

En pratique, pour éviter un dépassement de capacité, le vecteur y_k est normalisé à chaque étape : il est divisé par la valeur absolue de sa plus grande composante. Cette disposition limite aussi les erreurs de troncatures.

Le programme suivant démarre avec X(I) = 1 et s'arrête lorsque l'écart entre la valeur propre calculée au pas précédent et celle qui vient d'être calculée est inférieur à ε donné (1E-5 par exemple).


lecture de A
impression de A
Pour I = 1 à N
Xi = 1
Initialisation X
VP = 1.2345
Initialisation VP
Pour K = 1 à 40
Limitation du nombre d'itérations à 40
Pour I = 1 à N
Yi = 0
Calcul du Produit AX
Pour J = 1 à N
Yi = Yi + Aij * Xj
V1 = Y1 / X1
Z = 0
Pour I = 1 à N
Calcul de la norme max_i |Yi|
|Yi| > Z ?
Pour I = 1 à N
Normalisation de Yi
Xi = Yi / Z
VP = V1
imprimer V1 et Xi, i = 1 à N
fin
non
|VP-V1| < ε ?
oui

```
10 REM  CALCUL +GRANDE VALEUR PROPRE
20 DEFINT I-N
30 READ N
40 DIM A(N,N),X(N),Y(N)
50 FOR I=1 TO N:FOR J=1 TO N
60 READ A(I,J):NEXT J,I
70 CLS
75 LPRINT "MATRICE A"
80 FOR I=1 TO N:LPRINT:FOR J=1 TO N
90 LPRINT TAB((J-1)*7) A(I,J);:NEXTJ,I
130 FOR I=1 TO N:X(I)=1: NEXT I:X(1)=1
140 VP=1.23456789
150 FOR K=1 TO 40
160 FOR I=1 TO N
170 Y(I)=0
180 FOR J=1 TO N
190 Y(I)=Y(I)+A(I,J)*X(J)
200 NEXT J,I
210 V1=Y(1)/X(1)
215 Z=0:FOR I=1 TO N:IF ABS(Y(I))Z THEN Z=Y(I)
216 NEXT I
220 FOR I=1 TO N:X(I)=Y(I)/Z:NEXT I
230 IF ABS(VP-V1)(1.E-5 THEN 260
240 VP=V1:NEXT K
250 PRINT"NON CONVERGENCE"V1:END
260 LPRINT :LPRINT"PLUS GRANDE VALEUR PROPRE ";V1"
261 LPRINT "VECTEUR PROPRE ASSOCIE"
265 FOR I=1 TO N:LPRINT X(I):NEXT I
270 END
275 FOR I=1 TO N:FOR J=1 TO N
280 AT(I,J)=A(J,I):NEXT J,I
290 REM +GRANDE VALEUR PROPRE MATRICE TRANSPOSEE
300 DATA 3
310 DATA 2,-1,0,9,4,6,-8,0,-3
```

```
MATRICE A
 2      -1       0
 9       4       6
-8       0      -3
PLUS GRANDE VALEUR PROPRE  3.00001
VECTEUR PROPRE ASSOCIE
-.75
 .75
 1
```

DEFLATION DE LA MATRICE

Comme il vient d'être exposé, la méthode de la "puissance itérée" ne permet de calculer que la plus grande valeur propre et son vecteur associé.

Pour déterminer les autres valeurs propres, la matrice A est transformée en une autre matrice A_1 où les valeurs propres sont $\lambda_2 > \lambda_3 > \lambda_4 \ldots > \lambda_n > 0$.

C'est la méthode de déflation de la matrice :

soit V_1 le vecteur propre de A, associé à λ_1

et W_1 le vecteur propre de A^T associé à λ_1
(A^T est la matrice transposée de A, c'est-à-dire : $a^T_{ij} = a_{ij}$)

par définition :

$$AV_1 = \lambda_1 V_1 \text{ ou encore } (A - \lambda_1 I)\, V_1 = 0$$

soit :

$$A_1 = A - \frac{\lambda_1}{W_1^T . V_1} \cdot V_1 W_1^T$$

en multipliant à droite par V_1

$$\begin{aligned} A_1 V_1 &= AV_1 - \frac{\lambda_1}{W_1^T\, V_1} \cdot V_1 W_1^T\, V_1 \\ &= AV_1 - \lambda_1\, V_1 \cdot \frac{W_1^T\, V_1}{W_1^T\, V_1} \\ &= AV_1 - \lambda_1\, V_1 \\ &= 0 \end{aligned}$$

A_1 admet donc la valeur propre 0

si V_2 est le vecteur propre associé à λ_2, en multipliant A_1 par V_2

$$A_1 V_2 = AV_2 - \frac{\lambda_1}{W_1^T . V_1} V_1 W_1^T\, V_2$$

$$= AV_2 - \lambda_1\, V_1 \frac{W_1^T\, V_2}{W_1^T\, V_1}$$

mais, par définition : $\lambda_1\, W_1 = A^T\, W_1$

en transposant : $\lambda_1\, W_1^T = W_1^T\, A$

(on rappelle que $(AX)^T = X^T\, A^T$ et que $(A^T)^T = A$)

en multipliant par V_2 : $\lambda_1 W_1^T\, V_2 = W_1^T\, AV_2$

et par définition : $AV_2 = \lambda_2\, V_2$

d'où : $W_1^T \lambda_2 V_2 = \lambda_1 W_1^T V_2$

donc : $(\lambda_1 - \lambda_2) W_1^T V_2 = 0$

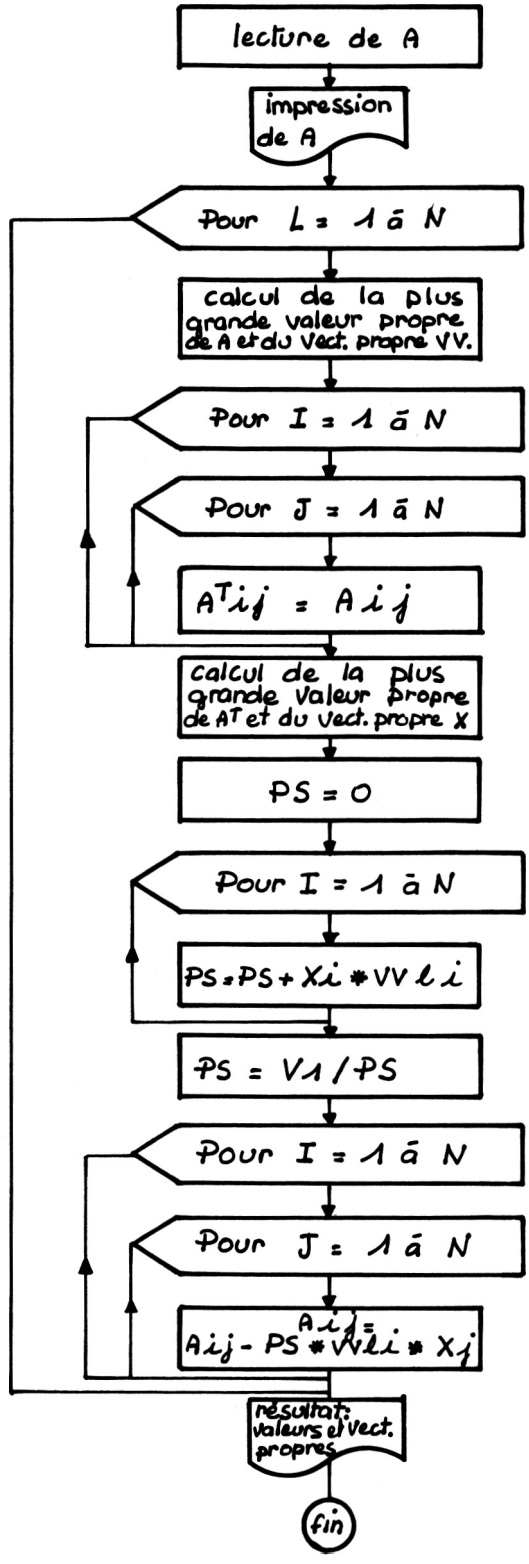
comme : $\lambda_1 \neq \lambda_2 \quad W_1^T . V_2 = 0$

d'où finalement : $A_1 V_2 = AV_2$

$= \lambda_2 V_2$

donc, λ_2 est la plus grande valeur propre de A_1 et alors, il est possible d'appliquer l'algorithme de la puissance itérée et ainsi de suite.

Les méthodes, "puissance itérée" et "déflation" doivent être appliquées avec prudence lorsqu'on n'est pas sûr du signe des valeurs propres et de la multiplicité éventuelle.


lecture de A
impression de A
Pour L = 1 à N
calcul de la plus grande valeur propre de A et du Vect. propre VV.
Pour I = 1 à N
Pour J = 1 à N
AT ij = A ij
calcul de la plus grande valeur propre de AT et du vect. propre X
PS = 0
Pour I = 1 à N
PS = PS + Xi * VV li
PS = V1 / PS
Pour I = 1 à N
Pour J = 1 à N
Aij - PS * VVli * Xj
résultat: valeurs et vect. propres
fin

```
10 REM CALCUL VALEURS ET VECTEURS PROPRES
20 DEFINT I-N
30 READ N
40 DIM A(N,N),AT(N,N),VV(N,N),X(N),Y(N),VD(N),W(N)
50 FOR I=1 TO N:FOR J=1 TO N
60 READ A(I,J):NEXT J,I
70 LPRINT"MATRICE"
80 FOR I=1 TO N:LPRINT:FOR J=1 TO N
90 LPRINT TAB((J-1)*7) A(I,J);:NEXTJ,I
100 LPRINT
110 FOR L=1 TO N
120 REM CALCUL + GRANDE VALEUR PROPRE
130 FOR I=1 TO N:X(I)=1: NEXT I:X(1)=1
140 VP=1.23456789
150 FOR K=1 TO 40
160 FOR I=1 TO N
170 Y(I)=0
180 FOR J=1 TO N
190 Y(I)=Y(I)+A(I,J)*X(J)
200 NEXT J,I
210 V1=Y(1)/X(1)
215 Z=0:FOR I=1 TO N:IF ABS(Y(I))>Z THEN Z=Y(I)
216 NEXT I
220 FOR I=1 TO N:X(I)=Y(I)/Z:NEXT I
230 IF ABS(VP-V1)<1.E-5 THEN 260
240 VP=V1:NEXT K
250 PRINT"NON CONVERGENCE"V1:END
260 PRINT:PRINT"VALEUR PROPRE"V1
261 PRINT"VECTEUR PROPRE"
265 FOR I=1 TO N:PRINT X(I):NEXT I
270 VD(L)=V1:VA=V1:FOR I=1 TO N:VV(L,I)=X(I):NEXT I
275 FOR I=1 TO N:FOR J=1 TO N
280 AT(I,J)=A(J,I):NEXT J,I
290 REM +GRANDE VALEUR PROPRE MATRICE TRANSPOSEE
300 FOR I=1 TO N:X(I)=1:NEXT I:X(1)=1
310 VP=1.23456
320 FOR K=1 TO 40
330 FOR I=1 TO N
340 Y(I)=0
350 FOR J=1 TO N
360 Y(I)=Y(I)+AT(I,J)*X(J)
370 NEXT J,I
380 V1=Y(1)/X(1)
385 Z=0:FOR I=1 TO N:IF ABS(Y(I))>Z THEN Z=Y(I)
386 NEXT I
```

```
390 FOR I=1 TO N:X(I)=Y(I)/Z:NEXT I
400 IF ABS(VP-V1)(1.E-5 THEN 430
410 VP=V1:NEXT K
420 GOTO250
430 IF ABS(VA-V1))3.E-5 THEN PRINT"ERREUR" VA,V1:END
440 REM DEFLATION
450 PS=0:FOR I=1 TO N:PS=PS+X(I)*VV(L,I):NEXT I
460 PS=V1/PS
470 FOR I=1 TO N:FOR J=1 TO N
480 A(I,J)=A(I,J)-PS*VV(L,I)*X(J)
490 NEXT J,I
500 NEXT L
510 LPRINT:FOR L=1 TO N:LPRINT "VALEUR PROPRE"VD(L)
515 FOR I=1 TO N
520 LPRINT VV(L,I):NEXT I:NEXT L
600 DATA 3
610 DATA 23,-4,16,40,-5,32,-20,4,-13
```

```
MATRICE
 23     -4       16
 40     -5       32
-20      4      -13
VALEUR PROPRE 2.99999
 .529411
 1
-.411764
VALEUR PROPRE 3
-.833336
-.166678
 1
VALEUR PROPRE-1
 .5
 1
-.5
```

ALGORITHME DE RUTISHAUSER

Cette méthode itérative permet de déterminer toutes les valeurs propres et est relativement simple et satisfaisante mais l'étude théorique est délicate et les démonstrations ne seront pas données complètement.

Le principe consiste à décomposer d'abord la matrice A en deux matrices L et R, L étant une matrice triangulaire inférieure, à diagonale unité et R une matrice triangulaire supérieure telles que :

$$A = L.R$$

puis à effectuer le produit matrice R.L qui donne une nouvelle matrice A_1 :

$$A_1 = RL$$

en général, $A_1 \neq A$, et on recommence :

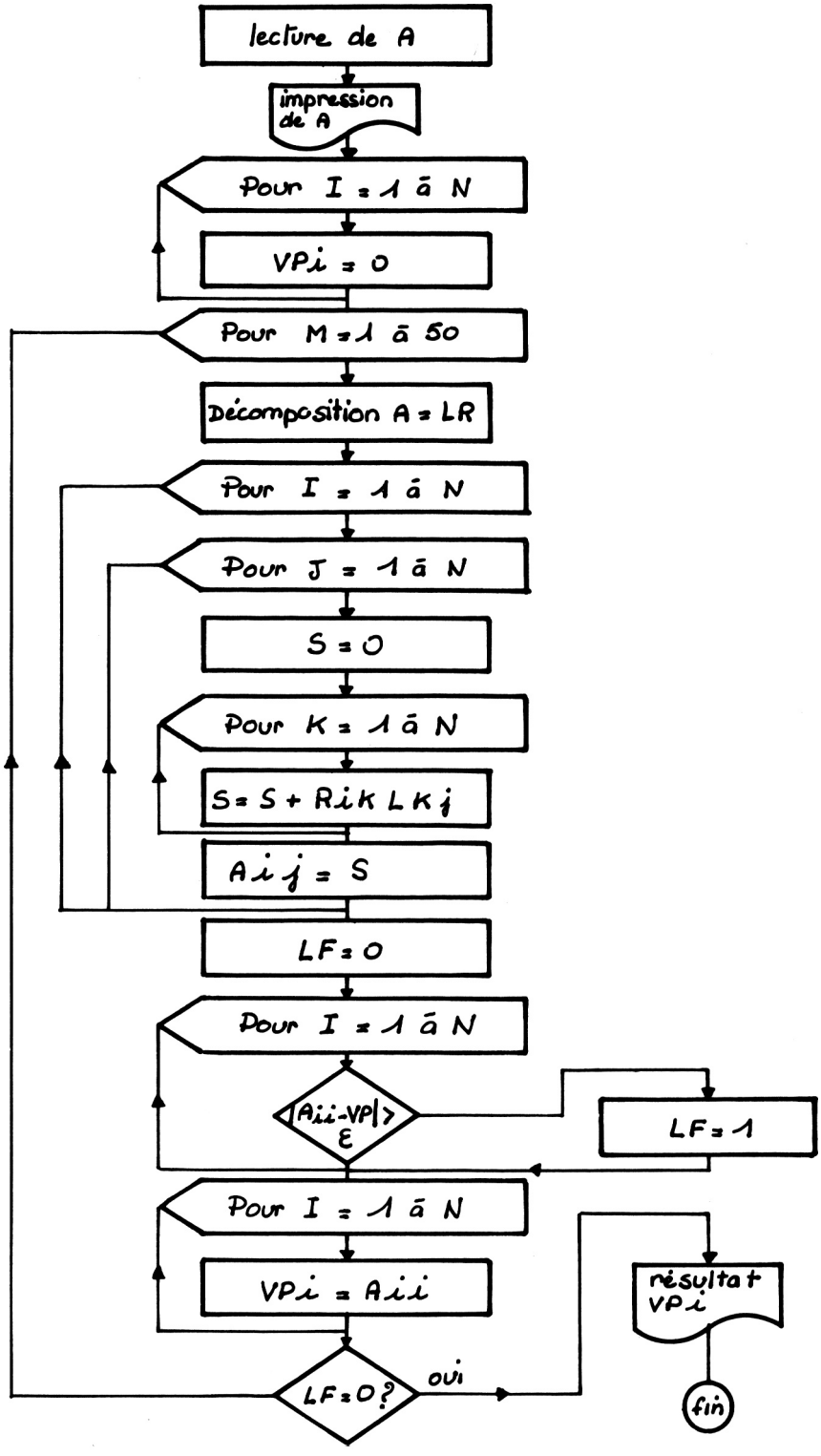
$$A_1 = L_1.R_1$$

puis $$A_2 = R_1.L_1$$

et ainsi de suite.

Ces opérations transforment A en une matrice qui devient triangulaire supérieure où les valeurs propres se trouvent sur la diagonale.

La méthode de décomposition est donnée au chapitre III, (page 56). Comme il a déjà été dit, la décomposition ne peut se faire que si les pivots successifs sont différents de 0.


lecture de A
impression de A
Pour I = 1 à N
VPi = 0
Pour M = 1 à 50
Décomposition A = LR
Pour I = 1 à N
Pour J = 1 à N
S = 0
Pour K = 1 à N
S = S + Rik LKj
Aij = S
LF = 0
Pour I = 1 à N
|Aii-VP| > ε
LF = 1
Pour I = 1 à N
VPi = Aii
LF = 0 ?
oui
résultat VPi
fin

```
10 REM CALCUL VALEURS PROPRES
20 DEFINT I-K,N
30 READ N
40 DIM A(N,N),L(N,N),R(N,N),VP(N)
45 FOR I=1 TO N:VP(I)=0:NEXT I
50 FOR I=1 TO N:FOR J=1 TO N
60 READ A(I,J):NEXT J,I
70 CLS
75 LPRINT "MATRICE A"
80 FOR I=1 TO N:LPRINT:FOR J=1 TO N
90 LPRINT TAB((J-1)*7) A(I,J);:NEXT J,I
95 FOR M=1 TO 50
100 FOR I=1 TO N:L(I,I)=1:NEXT I
110 FOR J=1 TO N:R(1,J)=A(1,J):NEXT J
120 FOR I=2 TO N:L(I,1)=A(I,1)/A(1,1):NEXT I
130 FOR K=2 TO N:FOR J=K TO N:TR=0
140 FOR I=1 TO K-1
145 TR=TR+R(I,J)*L(K,I):NEXT I
150 R(K,J)=A(K,J)-TR:NEXT J
160 IF K=N THEN 220
170 FOR I=K+1 TO N:TL=0
180 FOR J=1 TO K-1
190 TL=TL+L(I,J)*R(J,K):NEXT J
200 L(I,K)=(A(I,K)-TL)/R(K,K)
210 NEXT I
220 NEXT K
290 FOR I=1 TO N:FOR J=1 TO N:S=0
300 FOR K=1 TO N:S=S+R(I,K)*L(K,J):NEXT K
310 A(I,J)=S:NEXT J,I:LPRINT
315 PRINT:PRINT "MATRICE A=R*L"
316 PRINT M
320 FOR I=1 TO N:PRINT:FOR J=1 TO N
330 PRINT TAB((J-1)*7) A(I,J);:NEXT J,I
334 LF=0
335 FOR I=1 TO N
338 IF ABS(A(I,I)-VP(I)) > 1.E-4 THEN LF=1
340 NEXT I:FOR I=1 TO N:VP(I)=A(I,I):NEXT I
341 IF LF=0 THEN 345 ELSE NEXT M
342 PRINT"NON CONVERGENCE":END
345 LPRINT:LPRINT"VALEURS PROPRES":FOR I=1 TO N
346 LPRINT VP(I);:NEXT I:END
350 DATA 3
360 DATA 1,-3,-3,-1,-1,-3,1,5,7
```

```
MATRICE A
 1      -3      -3
-1      -1      -3
 1       5       7
VALEURS PROPRES
 3.99999   2.00001   1
```

METHODES ITERATIVES POUR LES SYSTEMES LINEAIRES

Les méthodes présentées au chapitre précédent sont dites "directes", la solution du système est obtenue en appliquant une fois l'algorithme. Lorsque les systèmes sont de grande taille, ce qui arrive très fréquemment lorsqu'il faut résoudre certains problèmes de mécanique des fluides, de transferts thermiques ..., les méthodes itératives sont plus efficaces : la précision est meilleure et les temps de calcul plus courts.

La matrice A du système linéaire :

$$Ax = b \qquad (x, b \text{ sont des vecteurs})$$

est remplacée par M-N, d'où :

$$(M-N)x = b$$

ou encore :

$$Mx = Nx+b$$

la solution est recherchée en utilisant la formule itérative :

$$Mx^{(p+1)} = Nx^{(p)}+b$$

qui peut s'écrire aussi :

$$x^{(p+1)} = M^{-1}Nx^{(p)}+M^{-1}b$$

$x^{(p+1)}$ est une meilleure approximation de la solution à partir du vecteur $x^{(p)}$ calculé à l'étape précédente.

Le calcul itératif commence en prenant un vecteur $x^{(0)}$ quelconque.

Sans faire toutes les démonstrations, examinons les conditions de convergence. L'erreur au bout de p itérations est notée $\varepsilon^{(p)} = (x^{(p)}-x)$, on cherche évidemment à ce que $\varepsilon^{(p)}$ deviennent le plus petit possible.

. $x^{(p)}$ est la valeur de la solution à l'itération p,

. x est la solution exacte.

La solution vérifie $Ax = b$ et $x = M^{-1}Nx+M^{-1}b$

d'où : $x^{(p+1)}-x = M^{-1}N(x^{(p)}-x)$

soit : $\varepsilon^{(p+1)} = B \ \varepsilon^{(p)}$

avec : $B = M^{-1}N$

les erreurs successives sont donc $\varepsilon^{(0)}$, $\varepsilon^{(1)}$, $\varepsilon^{(2)}$...

$$\varepsilon^{(1)} = B\,\varepsilon^{(0)}$$
$$\varepsilon^{(2)} = B\,\varepsilon^{(1)} = B^2\,\varepsilon^{(0)}$$

etc ...

d'où finalement $\varepsilon^{(p+1)} = B^p\,\varepsilon^{(0)}$

Ce qui veut dire que pour que la suite des vecteurs $x^{(1)}$, $x^{(2)}$, $x^{(3)}$... $x^{(p)}$ converge, c'est-à-dire que $x^{(p)}$ tende vers la solution x quand p augmente indéfiniment, il faut que B^p tende vers 0.

Si cette condition est remplie, la méthode converge quel que soit le vecteur de départ x.

Pour que B^p tende vers 0 quand p augmente, il faut que le module de la plus grande valeur propre de B soit inférieur à 1.

Les matrices M et N peuvent être choisies de différentes manières. Les méthodes classiques sont :

JACOBI

La matrice A est décomposée en trois matrices :

$$A = D-E-F$$

$$A = \begin{bmatrix} & & -F \\ & D & \\ -E & & \end{bmatrix}$$

$$M = D \quad , \quad N = E+F$$

donc : $$B = D^{-1}(E+F)$$

GAUSS-SEIDEL

En utilisant la même décomposition de A :

$$A = D-E-F$$

$$M = D-E \quad , \quad N = F$$

CONVERGENCE

On montre que pour qu'il y ait convergence de ces méthodes, il suffit que la matrice A soit symétrique, définie positive, c'est-à-dire que $Ax.x > 0$ quel que soit $x \neq 0$ (pour qu'une matrice symétrique soit définie positive, il faut et il suffit qu'elle ait toutes ses valeurs propres positives) ou que la matrice soit à diagonale strictement dominante :

$$|a_{ii}| > \sum_{j=1}^{n} |a_{ij}| \quad (j \neq i)$$

On montre également que la méthode de Gauss-Seidel converge deux fois plus vite que la méthode de Jacobi. En outre, elle ne nécessite le stockage que d'un vecteur au lieu de deux pour Jacobi.

ALGORITHME DE GAUSS-SEIDEL

La formule itérative de cette méthode est :

$$(D-E)\ x^{(p+1)} = F\ x^{(p)} + b$$

soit pour la ligne i :

$$a_{ii}\ x_i^{(p+1)} - a_{i1}\ x_1^{(p+1)} - a_{i2}\ x_2^{(p+1)} \ldots - a_{ii-1}\ x_{i-1}^{(p+1)} = a_{ii+1}\ x_{i+1}^{(p)} + a_{ii+2}\ x_{i+2}^{(p)} + \ldots + a_{in}\ x_n^{(p)} + b_i$$

en séparant les termes connus et inconnus :

$$a_{ii}\ x_i^{(p+1)} = a_{i1}\ x_1^{(p+1)} + \ldots + a_{ii-1}\ x_{i-1}^{(p+1)} + a_{ii+1}\ x_{i+1}^{(p)} + \ldots + a_{in}\ x_n^{(p)} + b_i$$

en effet, $x_1^{(p+1)}$, $x_2^{(p+1)} \ldots x_{i-1}^{(P+1)}$ sont calculées aux lignes i-1, i-2, ... 3, 2, 1.

On part d'un vecteur quelconque $x^{(0)}$ et on applique la relation, pour chaque ligne :

$$x_i^{(p+1)} = \frac{1}{a_{ii}} \left[\sum_{k=1}^{i-1} a_{ik}\ x_k^{(p+1)} + \sum_{k=i+1}^{n} a_{ik}\ x_k^{(p)} + b_i \right]$$

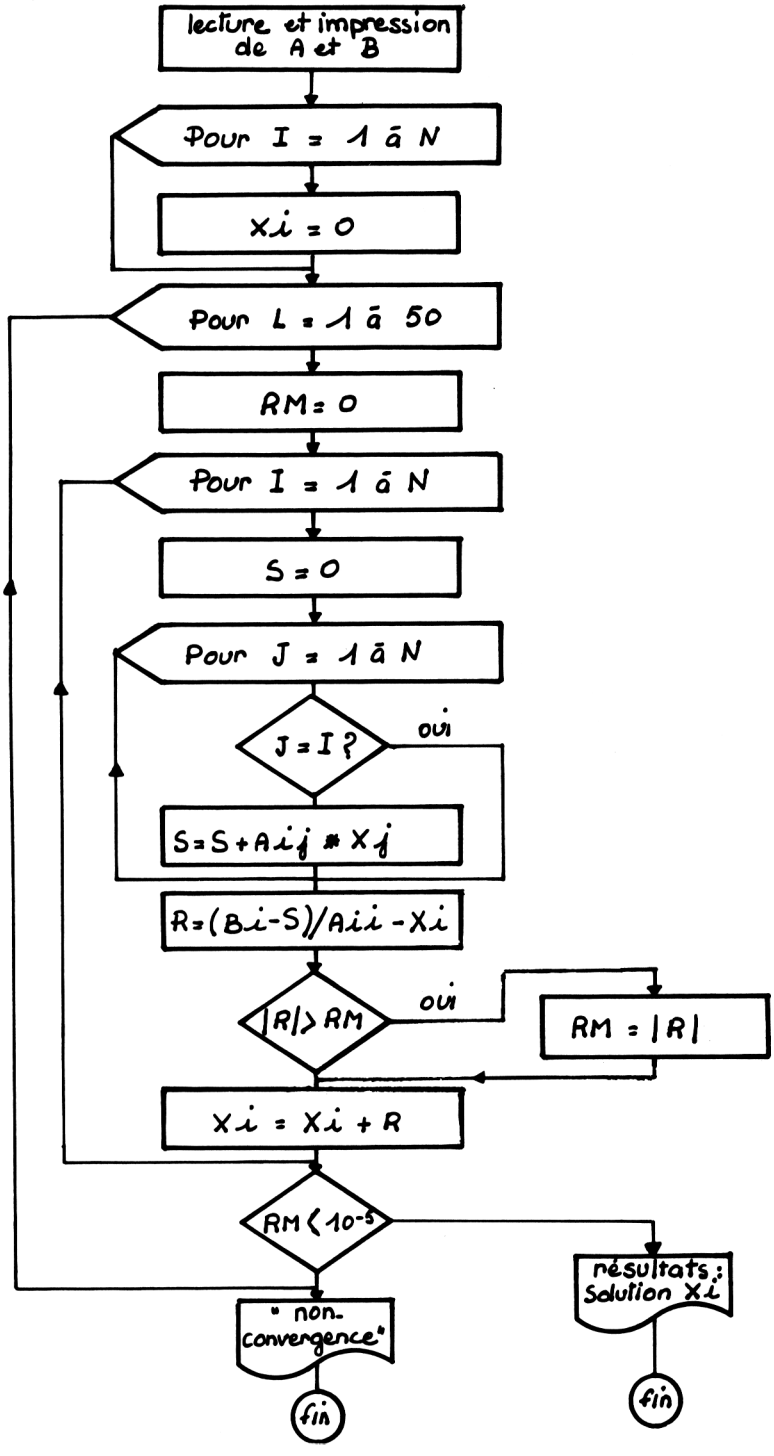
donc pour i de 1 à n, on recommence jusqu'à ce que le vecteur $x^{(p+1)}$ soit suffisamment proche de la solution. Pour cela, on pose :

$$x^{p+1} = x^{p} + R$$

R est appelé résidu, ce vecteur diminue à chaque étape. Pour effectuer le test, on utilise la norme :

$$\|R\| = \max_i |r_i|$$

c'est-à-dire la valeur absolue de la plus grande composante de R.



```
10 REM  SYSTEME LINEAIRE  A*X=B METHODE DE GAUSS-SEIDEL
20 DEFINT I-N
30 READ N
40 DIM A(N,N),X(N),B(N)
50 FOR I=1 TO N: FOR J=1 TO N
60 READ A(I,J):NEXT J,I
70 CLS
75 LPRINT "MATRICE A"
80 FOR I=1 TO N:LPRINT:FOR J=1 TO N
90 LPRINT TAB((J-1)*7) A(I,J);:NEXT J,I
100 FOR I=1 TO N:READ B(I):NEXT I
110 LPRINT
120 LPRINT "SECOND MEMBRE B"
130 FOR I=1 TO N:LPRINT "B( ";I;")=";B(I):NEXT I
140 FOR I=1 TO N:X(I)=0:NEXT I
150 FOR L=1 TO 50 :RM=0
160 FOR I=1 TO N
170 S=0
180 FOR J=1 TO N
190 IF J=I THEN 210
200 S=S+A(I,J)*X(J)
210 NEXT J
220 R=(B(I)-S)/A(I,I)-X(I)
230 IF ABS(R))RM THEN RM=ABS(R)
240 X(I)=X(I)+R
250 NEXT I
260 IF RM(1.E-5 THEN 285
270 NEXT L
280 PRINT"NON CONVERGENCE R=";R:END
285 LPRINT "SOLUTION DU SYSTEME AX=B":LPRINT
290 FOR I=1 TO N:LPRINT "X( ";I;")=";X(I):NEXT I
300 LPRINT "NB ITERATIONS ";L:END
310 DATA 3
320 DATA 4,1,2,1,3,1,1,2,5
330 DATA 16,10,12
340 DATA 1,0,0
```

```
MATRICE A
 4         1         2
 1         3         1
 1         2         5
SECOND MEMBRE B
B(  1 )= 16
B(  2 )= 10
B(  3 )= 12
SOLUTION DU SYSTEME AX=B
X(  1 )= 3
X(  2 )= 2
X(  3 )= 1
NB ITERATIONS  8
```

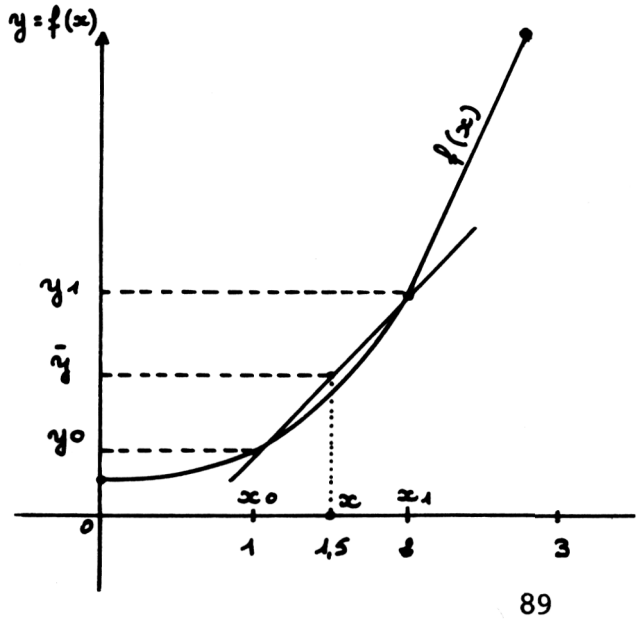
INTERPOLATION

POLYNOME DE LAGRANGE

L'interpolation linéaire permet de calculer approximativement une valeur intermédiaire dans une table.

La fonction y(x) est linéarisée dans l'intervalle : la courbe représentative est remplacée par une droite dans l'intervalle $[x_0, x_1]$.

Par exemple, trouver la valeur de y correspondant à x = 1.5 sachant que y(1) = 2.7128 et y(2) = 7.38906.



pour un intervalle $[x_0, x_1]$, les propriétés de la droite conduisent à

$$\frac{y_1-\bar{y}}{x_1-x} = \frac{y_1-y_0}{x_1-x_0}$$

d'où $\bar{y}=y_1-(x_1-x)\ \frac{y_1-y_0}{x_1-x_0}$

ou encore :

$$\frac{\bar{y}-y_0}{x-x_0} = \frac{y_1-y_0}{x_1-x_0}$$

d'où :

$$\bar{y}=y_0+(x-x_0)\frac{y_1-y_0}{x_1-x_0}$$

ici $\bar{y}(1.5) = 5.0567$

Ces deux expressions équivalentes peuvent s'écrire encore différemment :

$$\bar{y} = \frac{(x_1-x)y_0 - (x_0-x)y_1}{x_1-x_0} \qquad \text{forme d'Aitken}$$

ou $$\bar{y} = \frac{x-x_1}{x_0-x_1} y_0 + \frac{x-x_0}{x_1-x_0} y_1$$

Cette dernière forme est appelée polynôme de Lagrange. En effet, l'équation de la droite est un polynôme de degré 1.

En examinant le polynôme de Lagrange, on voit immédiatement que si $x=x_1$, le premier terme disparaît puisque $x_1-x_1=0$ et alors $y=y_1$, de même si $x=x_0$, $y=y_0$. Ce polynôme de Lagrange, de degré 1, est la somme de deux polynômes de degré 1.

L'approximation peut être meilleure si, au lieu d'utiliser un polynôme de degré 1 (une droite) passant par deux points, on utilise un polynôme de degré 2 (une parabole) passant par trois points ou d'une manière plus générale, un polynôme de degré n passant par n+1 points. Ce polynôme sera tel que si $x=x_0$, $y=y_0$, si $x=x_1$, $y=y_1$ etc ...

La formule générale d'interpolation de Lagrange devient :

$$\bar{y} = \frac{(x-x_1)(x-x_2)\ldots(x-x_n)}{(x_0-x_1)(x_0-x_2)\ldots(x_0-x_n)} y_0 + \frac{(x-x_0)(x-x_2)\ldots(x-x_n)}{(x_1-x_0)(x_1-x_2)\ldots(x_1-x_n)} y_1 +$$

$$+ \ldots + \frac{(x-x_0)(x-x_1)\ldots(x-x_{n-1})}{(x_{n-1}-x_0)(x_{n-1}-x_1)\ldots(x_{n-1}-x_n)} y_n$$

En posant $L_k(x) = (x-x_0)(x-x_1)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_n)$

alors :

$$P_n(x) = \sum_{k=0}^{n} y_k \frac{L_k(x)}{L_k(x_k)}$$

$P_n(x)$ est le polynôme d'interpolation de degré n, passant par n+1 points.

On montre que ce polynôme, pour un ensemble de couples de points (x_k,y_k), est unique.

Il est facile de vérifier, sous la forme de Lagrange, que si $x = x_j$, $P_n(x_j) = y_j$, en effet, $L_j(x_k) = 0$ pour $k \neq j$ et lorsque $k = j$, le polynôme de Lagrange se réduit donc à :

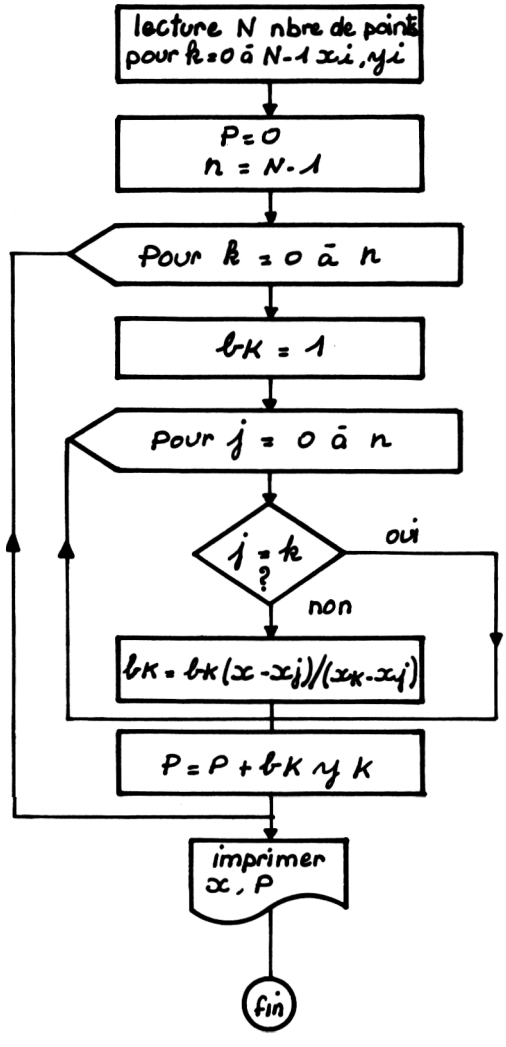
$$P_n(x_k) = \frac{L_k(x_k)}{L_k(x_k)} y_k = y_k$$

Cette formule d'interpolation s'applique pour des points quelconques (inégalement répartis) ; en échangeant les x et y, il est possible d'effectuer l'interpolation inverse : trouver $\bar{x}$ tel que $y = 4$ ($x_0=1, y_0=2.71828$; $x_1=2, y_1=7.38906$), on calcule à partir de :

$$\bar{x} = \frac{y-y_1}{y_0-y_1} x_0 + \frac{y-y_0}{y_1-y_0} x_1$$

on obtient $\bar{x} = 1.2744$.

Pour un ensemble d'abscisses x_k données, si on change les valeurs y_k correspondantes, il n'est pas nécessaire de refaire tous les calculs, les coefficients $\frac{L_k(x)}{L_k(x_k)}$ ne changent pas puisqu'ils ne dépendent pas de y et ainsi, pour des répartitions données, ils peuvent être tabulés (ou calculés et stockés).


lecture N nbre de points
pour k=0 à N-1 xi, yi
P=0
n = N-1
Pour k = 0 à n
bK = 1
Pour j = 0 à n
j = k ?
oui
non
bK = bK(x-xj)/(xK-xj)
P = P + bK yK
imprimer
x, P
fin

```
5 REM INTERPOLATION LAGRANGE
10 INPUT "NB DE POINTS";N1
15 LPRINT "NB DE POINTS ";N1
20 N=N1-1
30 DIM X(N),Y(N),B(N)
40 FOR K=0 TO N:PRINT K;:INPUT "X,Y";X(K),Y(K)
45 LPRINT "X=";X(K);"  Y=";Y(K):NEXT K
50 INPUT "X=";XX
60 P=0
70 FOR K=0 TO N:B(K)=1
80 FOR J=0 TO N
90 IF J=K THEN 110
100 B(K)=B(K)*(XX-X(J))/(X(K)-X(J))
110 NEXT J
120 P=P+B(K)*Y(K)
130 NEXT K
140 LPRINT "Y=";P;"POUR X=";XX
150 END
```

```
NB DE POINTS  2
X= 0   Y= 1
X= 1   Y= 2.71828
Y= 3.57742 POUR X= 1.5

NB DE POINTS  3
X= 0   Y= 1
X= 1   Y= 2.71828
X= 2   Y= 7.38906
Y= 4.68461 POUR X= 1.5

NB DE POINTS  4
X= 0   Y= 1
X= 1   Y= 2.71828
X= 2   Y= 7.38906
X= 3   Y= 20.0855
Y= 4.36754 POUR X= 1.5
```

FORMULE D'INTERPOLATION DE NEWTON

Le polynôme d'interpolation est unique ; $P_n(x)$ peut être écrit sous la forme classique :

$$P_n(x) = a_0x^n+a_1x^{n-1}+ \ldots +a_{n-1}x+a_n$$

sous cette forme, les coefficients (à expliciter) a_0, $a_1,a_2 \ldots a_{n-1},a_n$ dépendent évidemment de $x_0,x_1,x_2 \ldots x_n$ et $y_0,y_1, \ldots y_n$.

Pour calculer ces coefficients a_k, il suffit d'écrire le système d'équations linéaires correspondant à $P_n(x_i) = y_i$:

$$\begin{cases} a_0x_0^n+a_1x_0^{n-1}+ \ldots +a_{n-1}x_0+a_n = y_0 \\ a_0x_1^n+a_1x_1^{n-1}+ \ldots +a_{n-1}x_1+a_n = y_1 \\ \text{----------} \\ a_0x_n^n+a_1x_n^{n-1}+ \ldots +a_{n-1}x_n+a_n = y_n \end{cases}$$

Le déterminant de ce système est un déterminant de Vandermonde, qui est différent de zéro si les points x_i sont distincts.

Au paragraphe précédent, $P_n(x)$ a été développé sous la forme de Lagrange :

$$P_n(x) = \sum_{k=0}^{n} \frac{L_k(x)}{L_k(x_k)} y_k$$

pour l'interpolation, cette forme donne une évaluation plus rapide et plus précise.

Une autre forme, toujours équivalente, conduit à des calculs plus simples lorsque les points x_i sont équidistants, c'est la forme de Newton :

$$P_n(x) = c_0+c_1(x-x_0)+c_2(x-x_0)(x-x_1)+ \ldots$$

$$\ldots +c_n(x-x_0)(x-x_1)(x-x_2) \ldots (x-x_{n-1})$$

évidemment, les coefficients c_i sont tels que :

$$P_n(x_i) = y_i$$

Pour déterminer les coefficents ci, on utilise des formules de différence :

on pose : $\Delta y_i = y_{i+1} - y_i$

$$\Delta^2 y_i = \Delta(\Delta y_i) = \Delta(y_{i+1} - y_i)$$

$$= (y_{i+2} - y_{i+1}) - (y_{i+1} - y_i)$$

$$= y_{i+2} - 2y_{i+1} + y_i$$

$$\Delta^3 y_i = y_{i+3} - 3y_{i+2} + 3y_{i+1} - y_i$$

on retrouve ici les coefficients du binôme de Newton (ou triangle de Pascal) :

```
          1    1
        1    2    1
      1    3    3    1
    1    4    6    4    1
  ----------------------
```

Chaque coefficient est égal à la somme des deux coefficients, de la ligne précédente, situés de chaque côté.
Ex : 4 = 1+3 ; 6 = 3+3 ; 4 = 3+1.

Plus généralement :

$$\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i$$

En utilisant la formule de Newton :

$$\Delta^k y_0 = y_k - \binom{k}{1} y_{k-1} + \binom{k}{2} y_{k-2} \ldots + (-1)^k y_0$$

avec $\binom{k}{i} = \frac{k!}{i!(k-i)!}$ $\quad [k! = 1.2.3 \ldots k]$

Le calcul se fait facilement en utilisant un tableau de différences (ici, différence avant) :

i	x_i y_i	$\Delta y_i = y_{i+1}-y_i$	$\Delta^2 y_i = \Delta y_{i+1}-\Delta y_i$	$\Delta^3 y_i = \Delta^2 y_{i+1}-\Delta^2 y_i$	$\Delta^4 y_i = \Delta^3 y_{i+1}-\Delta^3 y_i$
0	x_0 y_0				
		Δy_0			
1	x_1 y_1		$\Delta^2 y_0$		
		Δy_1		$\Delta^3 y_0$	
2	x_2 y_2		$\Delta^2 y_1$		$\Delta^4 y_0$
		Δy_2		$\Delta^3 y_1$	
3	x_3 y_3		$\Delta^2 y_1$		
		Δy_3			
4	x_4 y_4				
⋮					

Pour montrer le déroulement des calculs, examinons le cas d'un polynôme du troisième degré :

$$P_3(x) = c_0+c_1(x-x_0)+c_2(x-x_0)(x-x_1)$$
$$+c_3(x-x_0)(x-x_1)(x-x_2)$$

on veut $P_3(x_i)=y_i$ pour $i=0,1,2,3$, les x_i sont espacés régulièrement de $h = x_{i+1}-x_i$

$$P_3(x_0) = c_0$$

$$P_3(x_1) = c_0+hc_1$$

$$P_3(x_2) = c_0+2hc_1+2h^2c_2$$

$$P_3(x_3) = c_0+3hc_1+6h^2c_2+6h^3c_3$$

en résolvant ce système :

$$c_0 = y_0$$

$$c_1 = \frac{y_1-y_0}{h} = \Delta y_0/h$$

$$c_2 = \frac{y_2-2y_1+y_0}{2h^2} = \Delta^2 y_0/2h^2$$

$$c_3 = \Delta^3 y_0/6h^3$$

d'où :

$$P_3(x) = y_0 + \frac{\Delta y_0}{h}(x-x_0) + \frac{\Delta^2 y_0}{2h^2}(x-x_0)(x-x_1) + \frac{\Delta^3 y_0}{6h^3}(x-x_0)(x-x_1)(x-x_2)$$

D'une manière générale, (en reprenant les calculs ci-dessus) :

$$P_n(x) = y_0 + \frac{\Delta y_0}{h}(x-x_0) + \frac{\Delta^2 y_0}{2h^2}(x-x_0)(x-x_1) + \ldots + \frac{\Delta^n y_0}{n!h^n}(x-x_0)(x-x_1)\ldots(x-x_{n-1})$$

avec $\Delta^k y_0 = y_k - \binom{k}{1} y_{k-1} + \binom{k}{2} y_{k-2} \ldots + (-1)^k y_0$

Apparemment, les calculs sont plus compliqués que pour la méthode de Lagrange, mais la programmation est assez simple. Par rapport à cette méthode, la méthode de Newton a l'avantage de prendre en compte des points supplémentaires sans refaire tous les calculs et en conséquence, il est possible d'ajuster facilement le degré du polynôme en fonction de la précision souhaitée.

La méthode de Newton a les inconvénients d'exiger le choix de points régulièrement espacés et de nécessiter la reprise de tous les calculs si les y_k changent. Enfin, l'interpolation inverse n'est pas possible.

Au lieu de prendre les différences "avant", il est possible de procéder avec des différences "arrière" :

$$\nabla y_n = y_n - y_{n-1}$$

$$\nabla^2 y_n = y_n - 2y_{n-1} + y_{n-2} \ldots$$

d'où :

$$\nabla^k y_n = y_n - \binom{k}{1} y_{n-1} + \binom{k}{2} y_{n-2} + \ldots + (-1)^k y_{n-k}$$

Exercice : *dans quel cas, les différences "arrière" sont préférables aux différences "avant" compte tenu de la possibilité d'arrêter les calculs lorsque la précision est atteinte ?*

En résumé, il faut d'abord construire la table des différences en utilisant les formules :

$$\Delta y_i = \Delta^1 y_i = y_{i+1} - y_i \quad \text{pour } i=0,n-1$$

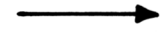
$$\Delta^k y_i = \Delta^{k-1} y_{i+1} - \Delta^{k-1} y_i \quad \text{pour } k \text{ de } 2 \text{ à } n \text{ et } i \text{ de } 0 \text{ à } n-k$$

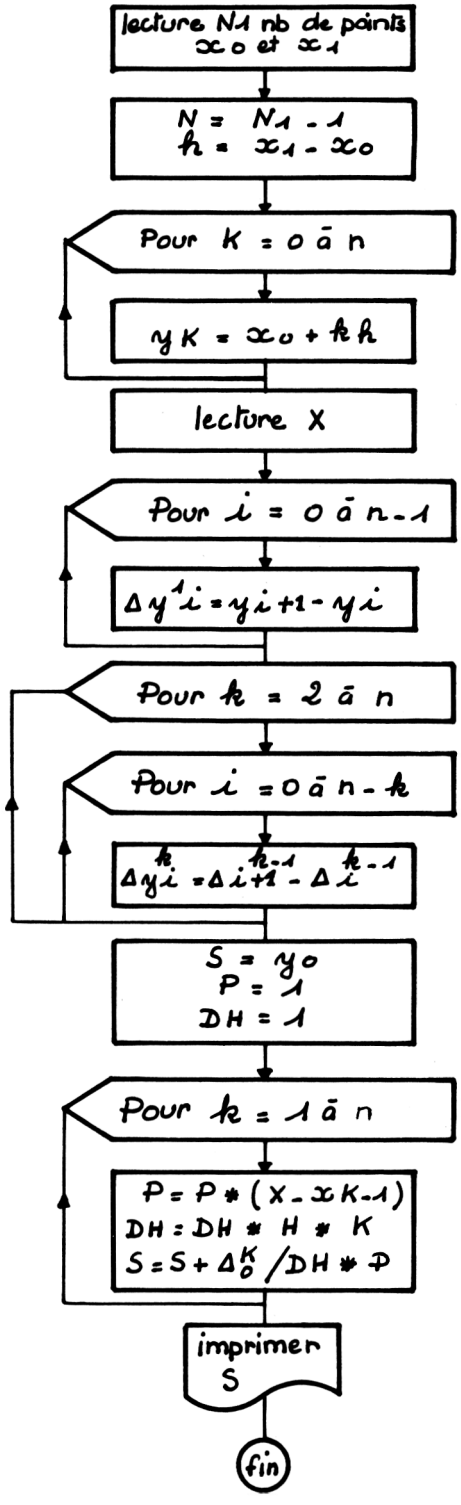
puis calculer le polynôme :

$$P_n(x) = y_0 + T_0 + T_1 \ldots + T_{n-1}$$

avec : $T_0 = \dfrac{\Delta^1 y_0}{h} (x-x_0)$

et : $T_k = \dfrac{\Delta^k y_0}{k! h^k} (x-x_0)(x-x_1) \ldots (x-x_k) \quad k=2 \text{ à } n-1$




lecture N1 nb de points
x0 et x1
N = N1 - 1
h = x1 - x0
Pour K = 0 à n
yK = x0 + kh
lecture X
Pour i = 0 à n-1
Δy1i = yi+1 - yi
Pour k = 2 à n
Pour i = 0 à n-k
Δkyi = Δk-1i+1 - Δk-1i
S = y0
P = 1
DH = 1
Pour k = 1 à n
P = P * (X - xK-1)
DH = DH * H * K
S = S + Δ0K / DH * P
imprimer
S
fin

```
10 INPUT "NB DE POINTS";N1
20 N=N1-1
30 DIM X(N),Y(N),DY(N,N)
40 INPUT "X(0),X(1)";X(0),X(1):H=X(1)-X(0)
45 FOR K=0 TO N
50 X(K)=X(0)+K*H:PRINT"X(";K;")=";X(K);
51 INPUT "Y=";Y(K):NEXT K
52 INPUT"X=";XX
55 FOR I=0 TO N-1
56 DY(1,I)=Y(I+1)-Y(I):NEXT I:IF N=1 THEN 60
57 FOR K=2 TO N:FOR I=0 TO N-K
58 DY(K,I)=DY(K-1,I+1)-DY(K-1,I)
59 NEXT I,K
60 FOR I=0 TO N-1:FOR K=1 TO N-I:PRINT DY(K,I);
61 NEXT K : PRINT
65 NEXT I
70 S=Y(0):P=1:DH=1
80 FOR K=1 TO N
90 P=P*(XX-X(K-1))
100 DH=DH*H*K
130 S=S+DY(K,0)/DH*P
140 NEXT K
145 PRINT
150 PRINT "Y= ";S ;"POUR X=";XX:END
```

Le programme qui vient d'être décrit a l'inconvénient de demander un tableau DY(N,N) pour le calcul des différences. Un examen des formules montre qu'il n'est pas nécessaire de conserver tout le tableau, il faut revoir le programme en conséquence.

Le programme suivant procède aussi à l'interpolation linéaire suivant la méthode de Newton, avec la formule des "différences avant", mais en utilisant YD(N) comme table de différences, contenant successivement $\Delta^k y_0$ en position 0, les autres valeurs sont celles, situées sur la ligne inférieure de la table des différences, qui sont nécessaires aux calculs suivants.

Comme dans de nombreux cas, il est recommandé d'exécuter manuellement le programme pas à pas, pour quelques étapes, pour se rendre compte du stockage des valeurs intermédiaires de calcul.

Enfin, si on veut, on peut arrêter le calcul facilement lorsque la précision désirée est atteinte.

Le programme utilisant les différences "arrière" est établi de la même manière.

```
5 REM INTERPOLATION NEWTON
10 INPUT "NB DE POINTS";N1
15 LPRINT "NB DE POINTS ";N1
20 N=N1-1
30 DIM X(N),Y(N),YD(N)
40 INPUT "X(0),X(1)";X(0),X(1):H=X(1)-X(0)
42 LPRINT "X(0),X(1) ";X(0);X(1)
45 FOR K=0 TO N
50 X(K)=X(0)+K*H:LPRINT"X(";K;")=";X(K);
51 INPUT"Y=";Y(K):LPRINT"Y=";Y(K):NEXT K
55 INPUT"X=";XX
60 FOR K=0 TO N:YD(K)=Y(K):NEXT K
70 S=Y(0):P=1:DH=1:H=X(1)-X(0)
80 FOR K=1 TO N
90 P=P*(XX-X(K-1))
100 DH=DH*H*K
110 FOR J=1 TO N
120 YD(J-1)=YD(J)-YD(J-1):NEXT J
130 S=S+YD(0)/DH*P
140 NEXT K
150 LPRINT "Y= ";S ;"POUR X=";XX:END
```

```
NB DE POINTS  2
X(0),X(1)  0  1
X( 0 )= 0 Y= 1
X( 1 )= 1 Y= 2.71828
Y=  3.57742 POUR X= 1.5

NB DE POINTS  3
X(0),X(1)  0  1
X( 0 )= 0 Y= 1
X( 1 )= 1 Y= 2.71828
X( 2 )= 2 Y= 7.38906
Y=  4.68461 POUR X= 1.5

NB DE POINTS  4
X(0),X(1)  0  1
X( 0 )= 0 Y= 1
X( 1 )= 1 Y= 2.71828
X( 2 )= 2 Y= 7.38906
X( 3 )= 3 Y= 20.0855
Y=  4.36754 POUR X= 1.5
```

```
NB DE POINTS  5
X(0),X(1)  0  1
X( 0 )= 0 Y= 1
X( 1 )= 1 Y= 2.71828
X( 2 )= 2 Y= 7.38906
X( 3 )= 3 Y= 20.0855
X( 4 )= 4 Y= 54.5982
Y=  4.57185 POUR X= 1.5

EXP(1.5)= 4.48169
```

ALGORITHME D'AITKEN

Une autre expression de l'unique polynôme d'interpolation est donnée par l'algorithme d'Aitken.

Soit $R_n(x)$ un polynôme d'interpolation construit à partir des points (x_0,y_0), (x_1,y_1) ... (x_{n-1},y_{n-1}), (x_a,y_a).

Lorsque $x = x_i$, le polynôme d'interpolation est égal à y_i :

$$R_n(x_i) = y_i \quad \text{pour } i=0 \text{ à } n-1$$

$$\text{et} \quad R_n(x_A) = y_A$$

Un autre polynôme d'interpolation $Q_n(x)$, construit à partir des points (x_0,y_0), (x_1,y_1) ... (x_{n-1},y_{n-1}), (x_B,y_B) possède les mêmes propriétés caractéristiques :

$$Q_n(x_i) = y_i \quad \text{pour } i=0 \text{ à } n-1$$

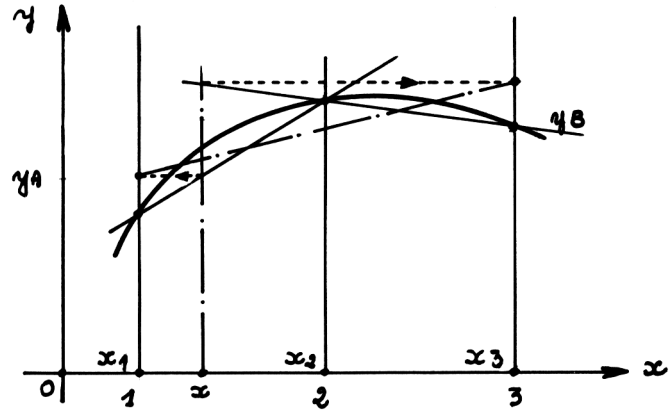
$$\text{et} \quad Q_n(x_B) = y_B$$

Le polynôme $P_n(x)$ obtenu par interpolation linéaire (suivant une des formules du polynôme) :

$$P_{n+1}(x) = \frac{(x_B-x)Q_n(x) - (x_A-x)R_n(x)}{x_B-x_A}$$

est le polynôme d'interpolation de degré n+1 passant par points (x_0,y_0), (x_1,y_1) ... (x_{n-1},y_{n-1}), (x_A,y_A), (x_B,y_B).

Examinons cette interpolation sur trois points : calcul de $\bar{y}$, valeur approchée de $y(x)$, ici $1 < x < 2$



l'interpolation linéaire entre x_1 et x_2 donne y_A :

$$y_A = \frac{(x2-x)y_1 - (x_1-x)y_2}{x_2-x_1}$$

et entre x_2 et x_3 :

$$y_B = \frac{(x_3-x)y_2 - (x_2-x)y_3}{x_3-x_2}$$

enfin $\bar{y}$, interpolation entre x_1 et x_3

$$\bar{y} = \frac{(x_3-x)y_A - (x_1-x)y_B}{x_3-x_1}$$

le graphique montre les trois étapes.

Les interpolations linéaires successives se calculent plus facilement en utilisant un tableau et les notations suivantes :

$$y_1^0 = \frac{y_0(x-x_1)-y_1(x-x_0)}{x_0-x_1}$$

$$y_1^1 = \frac{y_1(x-x_2) - y_2(x-x_1)}{x_1-x_2}$$

$$y_2^0 = \frac{y_1^0(x-x_2) - y_1^1(x-x_0)}{x_0-x_2}$$

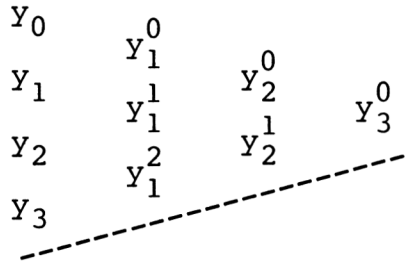
et d'une manière générale :

$$y_1^k = \frac{yk(x-xk+1) - yk+1(x-xk)}{x_k-x_{k+1}} \quad \text{pour } k=0 \text{ à } n-1$$

$$y_m^k = \frac{y_{m-1}^k(x-x_{k+m}) - y_{m-1}^{k+1}(x-xk)}{x_k-x_{k+m}} \quad \text{pour } m=2 \text{ à } n \text{ et } k=0 \text{ à } n-m$$

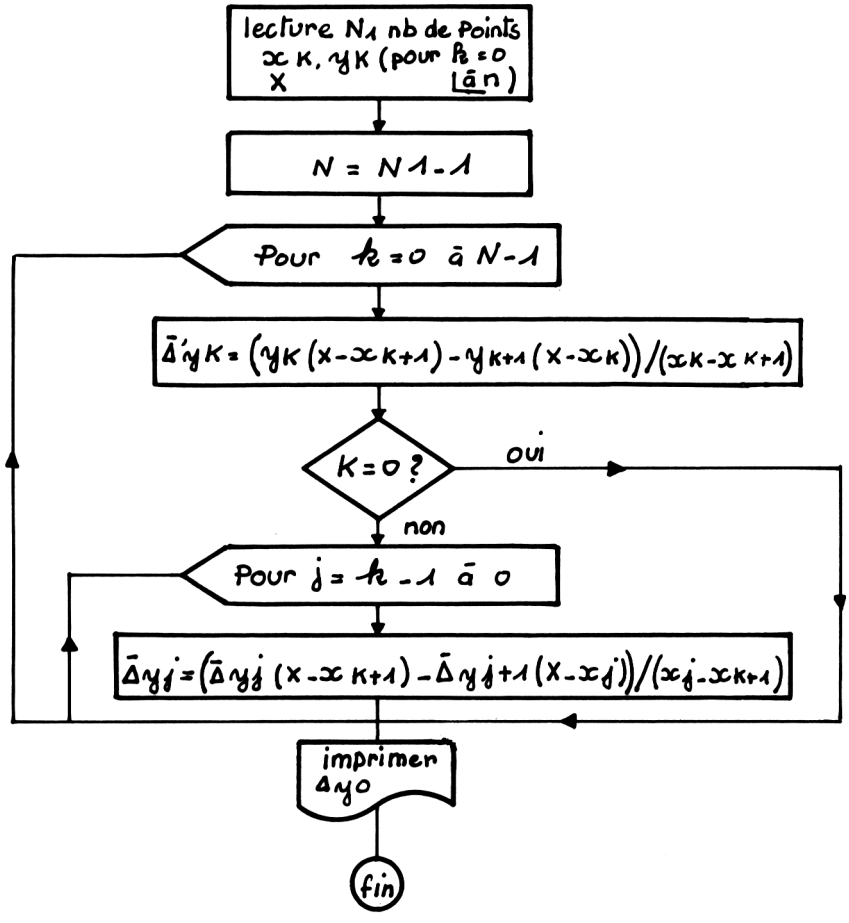
avec ces notations : $P_n(x) = y_n^0(x)$.

On dresse le tableau :



Cet algorithme se programme très facilement, les abscisses peuvent être choisies arbitrairement. L'interpolation inverse peut être faite et il est possible d'arrêter les calculs (nombre de points), lorsque la précision est atteinte.

Au lieu de dresser le tableau des interpolations linéaires successives, il suffit d'effectuer les calculs à partir de la dernière ligne.


lecture N1 nb de Points
xK, yK (pour k=0
X
à n)
N = N1-1
Pour k=0 à N-1
Δ'yK = (yK (X-xK+1) - yK+1 (X-xK))/(xK-xK+1)
K=0 ?
oui
non
Pour j = k-1 à 0
Δyj = (Δyj (X-xK+1) - Δyj+1 (X-xj))/(xj-xK+1)
imprimer
Δy0
fin

```
5 REM ALGORITHME D'AITKEN
10 INPUT "NB DE POINTS";N1
15 LPRINT "NB DE POINTS ";N1
20 N=N1-1
30 DIM X(N),Y(N),YD(N)
40 FOR K=0 TO N:PRINT K;:INPUT "X,Y";X(K),Y(K)
45 LPRINT "X= ";X(K);"  Y= ";Y(K):NEXT K
50 INPUT "X=";XX
60 FOR K=0 TO N-1
70 YD(K)=(Y(K)*(XX-X(K+1))-Y(K+1)*(XX-X(K)))/(X(K)-X(K+1))
80 IF K=0 THEN 120
90 FOR J=K-1 TO 0 STEP -1
100 YD(J)=(YD(J)*(XX-X(K+1))-YD(J+1)*(XX-X(J)))/(X(J)
110 NEXT J                                          -X(K+1))
120 NEXT K
130 LPRINT "Y=";YD(0);" POUR X=";XX
140 END
```

```
NB DE POINTS  2
X=  0    Y=  1
X=  1    Y=  2.71828
Y= 3.57742  POUR X= 1.5
```

```
NB DE POINTS  3
X=  0    Y=  1
X=  1    Y=  2.71828
X=  2    Y=  7.38906
Y= 4.68461  POUR X= 1.5
```

```
NB DE POINTS  4
X=  0    Y=  1
X=  1    Y=  2.71828
X=  2    Y=  7.38906
X=  3    Y=  20.0855
Y= 4.36754  POUR X= 1.5
```

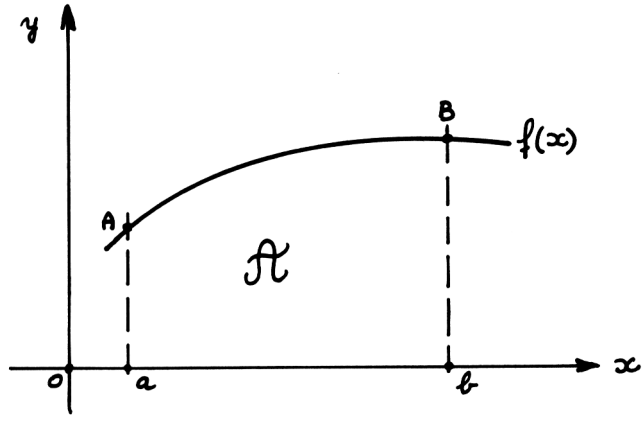
INTEGRATION

RAPPEL

Le but de ce chapitre est l'évaluation des intégrales définies, de la forme :

$$J = \int_a^b f(x).dx$$

où f(x) est une fonction continue sur l'intervalle [a,b] et peut alors être représentée par une courbe :



On rappelle que l'intégrale définie J, qui est alors un scalaire (nombre) est proportionnelle à l'aire $\mathcal{A}$, dont le périmètre est aABba. Si on définit un ensemble de points qui divisent l'intervalle [a,b] en n parties, alors J est la limite (quand le nombre de points augmente indéfiniment) de la somme des rectangles $(x_{i+1}-x_i).f(x_i)$, c'est-à-dire :

$$J = \lim_{n \to \infty} \sum_{i=0}^{n} f(x_i)(x_{i+1}-x_i)$$

En prenant n grand, on devrait approcher J ; mais, les erreurs de troncature s'accumulent et il faut utiliser des méthodes plus fines.

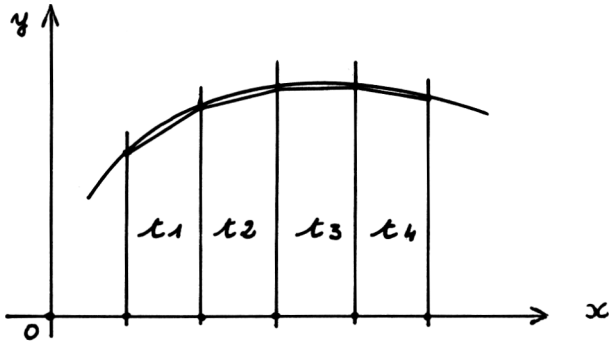
Si les points x_i sont équidistants, tels que $x_{i+1}-x_i=h$, la formule devient :

$$S_n = \sum_{i=0}^{n} h\, f(x_i) \quad \text{avec } h = \frac{b-a}{n}$$

en posant $f(x_0) = f_0$, $f(x_1) = f_1$ ..., la formule devient :

$$S_n = h \sum_{i=0}^{n} f_i$$

Une méthode, qui n'est pas beaucoup plus compliquée, donne une précision nettement meilleure : au lieu de décomposer la surface en rectangles, on utilise des trapèzes. Par exemple, pour quatre intervalles (n=4) :



$S_n=t_1+t_2+t_3+t_4$

$$= \sum_{i=1}^{4} t_i$$

où t_i sont les surfaces élémentaires

$t_i = (f(x_i)+f(x_{i-1}))/2*h$

$$= \frac{h}{2} f_i+f_{i-1}$$

d'où $S_4 = \frac{h}{2}((f_0+f_1)+(f_1+f_2)+(f_2+f_3)+(f_3+f_4))$

ou encore $S_4 = \frac{h}{2} (f_0+2f_1+2f_2+2f_3+f_4)$

et d'une manière générale :

$$S_n = \frac{b-a}{n} (f_0+2f_1+2f_2+ \ldots +2f_{n-1}+f_n)$$

$$\text{ou } S_n = \frac{b-a}{n} \left\{ f_0 + 2 \sum_{i=1}^{n-1} f_i + f_n \right\}$$

Cette formule est de la forme :

$$S_n = \sum_{i=0}^{n} A_i \, f_i$$

Les trois méthodes qui sont présentées utilisent ce type de formule :

- méthode des coefficients indéterminés pour des points x_i arbitraires
- méthode pour des points x_i équidistants (méthode des trapèzes et méthode de Simpson)
- méthode de Gauss pour des points xi choisis "au mieux", donnant ainsi une meilleure précision

METHODE DES COEFFICIENTS INDETERMINES

On recherche ici des coefficients de pondération A_i (i=0 à n) tels que l'erreur E donnée par :

$$E = \int_a^b f(x)dx - \left[A_0 \, f(x_0) + A_1 f(x_1) + \ldots + A_n \, f(x_n) \right]$$

soit nulle pour l'ensemble de points x_i (i=0 à n) distincts, quand f(x) est un polynôme $P_n(x)$ de degré inférieur ou égal à n.

En prenant successivement comme polynômes $1, x, x^2, x^3 \ldots$ x_n pour f(x), l'intégration conduit à un système d'équations linéaires : (a=x0, b=xn)

pour $f(x) = 1$: $\int_a^b 1\,dx = A_0 1 + A_1 1 + \ldots + A_n 1 = b-a$

pour $f(x) = x$: $\int_a^b x\,dx = A_0 x_0 + A_1 x_1 + \ldots + A_n x_n = \frac{b^2-a^2}{2}$

pour $f(x) = x^2$: $\int_a^b x^2 dx = A_0 x_0^2 + A_1 x_1^2 + \ldots + A_n x_n^2 = \frac{b^3-a^3}{3}$

---

pour $f(x) = x^n$: $\int_a^b x^n dx = A_0 x_0^n + A_i x_1^n + \ldots + A_n x_n^n = \frac{b^{n+1}-a^{n+1}}{n+1}$

Ce système linéaire, pour calculer les coefficients $A_0, A_1, A_2 \ldots A_n$ s'écrit, sous forme matricielle.

$$\begin{vmatrix} 1 & 1 & 1 & \ldots & 1 \\ x_0 & x_1 & x_2 & \ldots & x_n \\ x_0^2 & x_1^2 & x_2^2 & \ldots & x_n^2 \\ \vdots & \vdots & \vdots & & \vdots \\ x_0^n & x_1^n & x_2^n & \ldots & x_n^n \end{vmatrix} \begin{vmatrix} A_0 \\ A_1 \\ A_2 \\ \vdots \\ A_n \end{vmatrix} = \begin{vmatrix} b-a \\ \frac{b^2-a^2}{2} \\ \frac{b^3-a^3}{3} \\ \\ \frac{b^{n+1}-a^{n+1}}{n+1} \end{vmatrix}$$

Le déterminant de cette matrice n'est jamais nul si les points x_i sont distincts (déterminant de Vandermonde), le système admet donc une solution.

L'exemple numérique suivant montre la succession des différents calculs :

Soit la fonction $f(x) = \text{ch } x$ (équation de la chaînette) et les points : $y=(e^{+x}+e^{-x})/2$

x	y
0	1
1	1.5431
2	3.7622

calculer la valeur de $J=\int_0^2 \text{ch } x\,dx$, donc $\int_0^2 \text{ch } x\,dx \simeq$

$A_0\ f(0) + A_1\ f(1) + A_2\ f(2)$

en reprenant les calculs pour $1, x, x^2$:

$$\begin{vmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 1 & 4 \end{vmatrix} \begin{vmatrix} A_0 \\ A_1 \\ A_2 \end{vmatrix} = \begin{vmatrix} 2-0 \\ \frac{4-0}{2} \\ \frac{8-0}{3} \end{vmatrix}$$

la solution de ce système linéaire de trois équations à trois inconnues A_0,A_1 et A_2 est : $A_0 = 1/3$, $A_1 = 4/3$, $A_2 = 1/3$

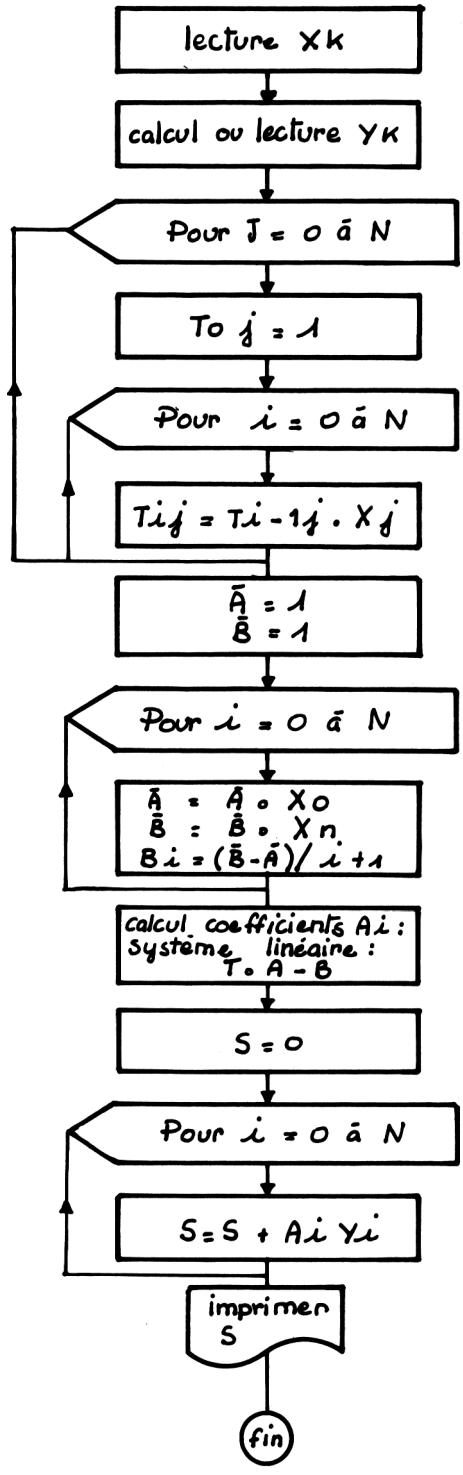
d'où $\int_0^2 \text{ch}\, x\, dx \simeq \frac{1}{3} \times 1 + \frac{4}{3} \times 1.5431 + \frac{1}{3} \times 3.7612 = 3.6449$

cette intégrale se calcule facilement $\int_0^2 \text{ch}\, x\, dx = \left[\text{sh}\, x\right]_0^2$

soit 3.6269 ; l'erreur est donc égale à :

$$3.6269-3.6449 = -0.018$$

Le programme de calcul s'établit en reprenant le programme de résolution des systèmes linéaires (voir chapitre III). On montre qu'il est préférable de ne pas prendre trop de points. Lorsque ces points sont équidistants, ce procédé conduit aux formules de Newton-Côtes.


lecture Xk
calcul ou lecture Yk
Pour J = 0 à N
To j = 1
Pour i = 0 à N
Tij = Ti-1j . Xj
Ā = 1
B̄ = 1
Pour i = 0 à N
Ā = Ā ∘ X0
B̄ = B̄ ∘ Xn
Bi = (B̄-Ā)/ i+1
calcul coefficients Ai :
Système linéaire :
T ∘ A - B
S = 0
Pour i = 0 à N
S = S + Ai Yi
imprimer
S
fin

```
10 REM INTEGRATION  ** COEFFICIENTS INDETERMINES **
20 READ N:LPRINT "NB DE POINTS";N
30 DIM X(N),Y(N),A(N),T(N,N),B(N)
40 FOR K=0 TO N:READ X(K):NEXT K
50 FOR K=0 TO N:XX=X(K):GOSUB 1000:Y(K)=YY:NEXT K
55 LPRINT "Y=1/X"
60 CLS:FOR K=0 TO N:LPRINT "X=";X(K),"Y=";Y(K):NEXT K
70 REM CALCUL COEFFICIENT DU SYSTEME LINEAIRE
80 FOR J=0 TO N:T(0,J)=1:FOR I=1 TO N
90 T(I,J)=T(I-1,J)*X(J):NEXT I,J
100 REM COEFFICIENTS DU SECOND MEMBRE
110 AA=1:BB=1:FOR I=0 TO N
120 AA=AA*X(0):BB=BB*X(N)
130 B(I)=(BB-AA)/(I+1)
140 NEXT I
150 REM CALCUL COEFFICIENTS A(I)
160 REM TRIANGULARISATION
170 FOR K=0 TO N-1:FOR I=K+1 TO N
180 B(I)=B(I)-T(I,K)/T(K,K)*B(K)
190 FOR J=K+1 TO N:T(I,J)=T(I,J)-T(I,K)/T(K,K)*T(K,J):
200 NEXT K                                          NEXT J,I
210 REM SOLUTION DU SYSTEME (MATRICE TRIANGULAIRE)
220 A(N)=B(N)/T(N,N)
230 FOR I=N-1 TO 0 STEP -1 :S=0
240 FOR K=I+1 TO N:S=S+T(I,K)*A(K):NEXT K
250 A(I)=(B(I)-S)/T(I,I):NEXT I
260 LPRINT "COEFFICIENTS A":FOR I=0 TO N:LPRINT"A(";I;")
270 NEXT I                                              =";A(I)
280 REM CALCUL DE L'INTEGRALE
290 S=0:FOR I=0 TO N:S=S+A(I)*Y(I):NEXT I
295 LPRINT "INTEGRALE DE Y=1/X  (  LOG(X)  )"
300 LPRINT "J SUR 1 , 4 =";S:END
1000 YY=1/XX:RETURN
3000 DATA 6
3001 DATA 1,1.5,2,2.5,3,3.5,4
```

```
NB DE POINTS 6
Y=1/X
X= 1              Y= 1
X= 1.5            Y= .666667
X= 2              Y= .5
X= 2.5            Y= .4
X= 3              Y= .333333
X= 3.5            Y= .285714
X= 4              Y= .25
COEFFICIENTS A
A( 0 )= .14648
A( 1 )= .771142
A( 2 )= .0970869
A( 3 )= .970622
A( 4 )= .0969849
A( 5 )= .771224
A( 6 )= .14646
INTEGRALE DE Y=1/X  (  LOG(X)  )
J SUR 1 , 4 = 1.38666
```

FORMULES DE NEWTON-COTES ET METHODES COMPOSITES

Maintenant, l'intervalle $[a,b]$ est obligatoirement divisé en n parties égales. Le "pas" d'intégration est :

$$h = \frac{b-a}{n}$$

alors, la formule d'intégration s'écrit :

$$\int_a^b f(x).dx \simeq (b-a) \sum_{j=0}^{n} B_j^{(n)} \; f(a+jh)$$

. si n = 1

$$\int_a^b f(x).dx \simeq (b-a) \left[\frac{1}{2} f(a) + \frac{1}{2} f(b)\right]$$

on retrouve la formule des trapèzes

. si n = 2

$$\int_a^b f(x).dx \simeq (b-a) \left[\frac{1}{6} f(a) + \frac{4}{6} f \frac{a+b}{2} + \frac{1}{6} f(b)\right]$$

c'est la formule de Simpson.

On démontre que $B_j^{(n)} = B_{n-j}^{(n)}$ (symétrie) et $\sum B_j^{(n)} = 1$; ces coefficients $B_j^{(n)}$ sont, en quelque sorte, des coefficients de pondération.

pour n = 3, $B_0^{(3)} = \frac{1}{8}$ $B_1^{(3)} = \frac{3}{8}$ $B_2^{(3)} = \frac{3}{8}$ $B_3^{(3)} = \frac{1}{8}$

pour n = 4, $B_0^{(4)} = \frac{7}{90}$ $B_1^{(4)} = \frac{32}{90}$ $B_2^{(4)} = \frac{12}{90}$ $B_3^{(4)} = \frac{32}{90}$

$$B_4^{(4)} = \frac{7}{90}$$

etc ...

Comme il a déjà été mentionné, il n'est pas recommandé de prendre n élevé : ces formules ne sont pas stables : les erreurs ou perturbations sont amplifiées.

C'est pourquoi, il est préférable d'utiliser des formules composites où les formules de Newton-Côtes sont appliquées à des sous intervalles. On obtient ainsi :

FORMULE DES TRAPEZES

$$h = \frac{b-a}{n}$$

$$\int_a^b f(x)dx = \sum_{k=0}^{n-1} \int_{a+kh}^{a+(k+1)h} f(x)dx$$

$$\simeq \sum_{k=0}^{n-1} \left[h \frac{1}{2} f(a+kh) + \frac{1}{2} f\ (a+(k+1)h) \right]$$

donc :

$$\int_a^b f(x)dx \simeq h \left[\frac{1}{2} f(a) + f(a+h) + \ldots + f(a+(n-1)h) + \frac{1}{2} f(b) \right]$$

FORMULE DE SIMPSON

$$h = \frac{b-a}{n} \quad \underline{\text{n pair}}$$

On choisit des intervalles de longueur 2h :

(a, a+2h), (a+2h, a+4h) ... (a+(n-2)h, a+nh)

$$\int_a^{a+2h} f(x)dx \simeq 2h \left[\frac{1}{6} f(a) + \frac{4}{6} f(a+h) + \frac{1}{6} f(a+2h) \right]$$

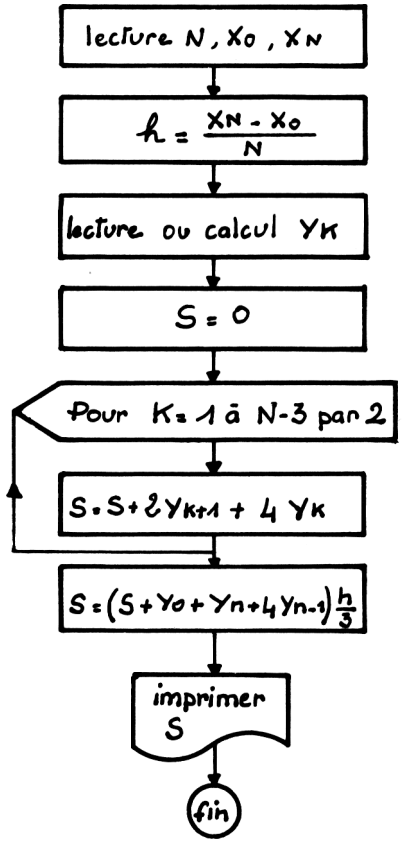
$$\int_{a+2h}^{a+4h} f(x)dx \simeq 2h \left[\frac{1}{6} f(a+2h) + \frac{4}{6} f(a+3h) + \frac{1}{6} f(a+4h) \right]$$

$$\int_{a+(n-2)h}^{b} f(x)dx \simeq 2h \left[\frac{1}{6} f(a+(n-2)h) + \frac{4}{6} f(a+(n-1)h) + \frac{1}{6} f(b)\right]$$

et en additionnant :

$$\int_{a}^{b} f(x)dx \simeq \frac{h}{3} \{f(a)+f(b)+2\left[f(a+2h)+f(a+4h)+..+f(a+(n-2)h)\right] +4\left[f(a+h) +f(a+3h)+..+f(a+(n-1)h)\right]$$

L'organigramme n'est donné que pour la méthode de Simpson, celui correspondant à la méthode des trapèzes est très proche.



```
10 REM INTEGRATION  ** FORMULE DES TRAPEZES **
20 READ N
30 DIM X(N),Y(N)
40 READ X(0),X(N):H=(X(N)-X(0))/N
45 LPRINT"Y=1/X
50 FOR K=0 TO N:X(K)=X(0)+K*H:XX=X(K):GOSUB 1000
55 Y(K)=YY:NEXT K
60 CLS:FOR K=0 TO N:LPRINT "X=";X(K),"Y=";Y(K):NEXT K
70 S=0:FOR K=1 TO N-1:S=S+Y(K):NEXT K
80 S=(S+Y(0)/2+Y(N)/2)*H
90 LPRINT "INTEGRALE DE Y=1/X  INTERVALLE 1,4  "
100 LPRINT " (  SOIT  J=LOG(4)   )"
300 LPRINT "J=";S:END
1000 YY=1/XX:RETURN
3000 DATA 6
3001 DATA 1,4
```

```
Y=1/X
X= 1              Y= 1
X= 1.5            Y= .666667
X= 2              Y= .5
X= 2.5            Y= .4
X= 3              Y= .333333
X= 3.5            Y= .285714
X= 4              Y= .25
INTEGRALE DE Y=1/X  INTERVALLE 1,4
 (  SOIT  J=LOG(4)   )
J= 1.40536
```

```
10 REM INTEGRATION  ** FORMULE DE SIMPSON **
15 READ N%
20 IF FIX(N%/2)*2<>N% THEN PRINT"N IMPAIR":END ELSE N=N%
30 DIM X(N),Y(N)
40 READ X(0),X(N):H=(X(N)-X(0))/N
45 LPRINT "Y=1/X"
50 FOR K=0 TO N:X(K)=X(0)+K*H:XX=X(K):GOSUB 1000
55 Y(K)=YY:NEXT K
60 CLS:FOR K=0 TO N:LPRINT "X=";X(K),"Y=";Y(K):NEXT K
70 S=0:FOR K=1 TO N-3 STEP 2:S=S+2*Y(K+1)+4*Y(K):NEXT K
80 S=(S+Y(0)+Y(N)+4*Y(N-1))*H/3
90 LPRINT "INTEGRALE DE Y=1/X   INTERVALLE ";X(0);X(N)
100 LPRINT " (  SOIT J=LOG(4)  )"
300 LPRINT "J=";S:END
1000 YY=1/XX:RETURN
3000 DATA 10
3001 DATA 1,4
```

```
Y=1/X
X= 1              Y= 1
X= 1.3            Y= .769231
X= 1.6            Y= .625
X= 1.9            Y= .526316
X= 2.2            Y= .454545
X= 2.5            Y= .4
X= 2.8            Y= .357143
X= 3.1            Y= .322581
X= 3.4            Y= .294118
X= 3.7            Y= .27027
X= 4              Y= .25
INTEGRALE DE Y=1/X   INTERVALLE  1  4
 (  SOIT J=LOG(4)  )
J= 1.38652
```

METHODE DE GAUSS

Cette méthode, encore de la forme :

$$\int_a^b f(x)\,dx \simeq \sum_{i=0}^{n} A_i\, f(x_i)$$

appliquée sur n+1 points x_i $(x_0, x_1, x_2 \ldots x_{n-1}, x_n)$ choisis judicieusement, donne une erreur nulle si f(x) est un polynôme $P_{2n+1}(x)$ de degré 2n+1. C'est-à-dire :

$$\int_a^b P_{2n+1}(x)\,dx - \sum_{i=0}^{n} A_i\, f(x_i) = 0$$

Il faut, dans le cas présent, déterminer non seulement les coefficients A_i, mais aussi les abscisses particulières x_i ; le problème se ramène donc à déterminer n+1 coefficients A_i et n+1 abscisses x_i, soit au total 2n+2 inconnues. On remarquera que le polynôme P_{2n+1} possède 2n+2 coefficients $(a_0, a_1, a_2 \ldots a_{2n}, a_{2n+1})$.

Le calcul sera fait sur l'intervalle $[-1,+1]$, mais avec un changement de variable très simple, on peut évaluer les intégrales sur n'importe quel intervalle $[a,b]$.

Le polynôme P_{2n+1} peut s'écrire sous la forme :

$$P_{2n+1} = P_n(x) + R_{n+1}(x).Q_n(x)$$

où $P_n(x)$ est le polynôme d'interpolation de degré n, construit sur l'ensemble des points x_i, donc :

$$P_n(x_i) = f(x_i)$$

ou plus simplement :

$$P_n(x_i) = f_i \qquad (i=0 \text{ à } n)$$

$R_{n+1}(x)$ est un polynôme de degré n+1 (qui, on le verra plus loin, a pour racines $x_0, x_1, \ldots x_n$), $Q_n(x)$ est un polynôme de degré au plus égal à n.

En intégrant sur l'intervalle $[-1,+1]$, on obtient :

$$\int_{-1}^{+1} P_{2n+1}(x)\,dx = \int_{-1}^{+1} P_n(x)\,dx + \int_{-1}^{+1} R_{n+1}(x).Q_n(x)\,dx$$

$$= \sum_{i=0}^{n} A_i\, P_n(x_i) + \int_{-1}^{+1} R_{n+1}(x).Q_n(x)\,dx$$

Pour avoir $\int_{-1}^{+1} P_{2n+1}(x)\,dx = \sum_{i=0}^{n} A_i\, P_n(x_i)$, il faut donc que la dernière intégrale $\int_{-1}^{+1} R_{n+1} Q_n\,dx$ soit nulle.

Si $R_0, R_1, R_2, \ldots R_n$ sont des polynômes de degré 0,1,2, ...n, alors le polynôme $Q_n(x)$ peut s'écrire sous la forme :

$$Q_n(x) = a_0\, R_0(x) + a_1\, R_1(x) + \ldots + a_n\, R_n(x)$$

Compte tenu de cette décomposition, pour que l'intégrale $\int_{-1}^{+1} R_{n+1}.Q_n\,dx$ soit nulle, il suffit que :

$$\int_{-1}^{+1} R_i(x)\; R_{n+1}(x)\,dx = 0 \qquad \text{pour tout } i \leqslant n$$

c'est la définition des polynômes orthogonaux de Legendre R_k.

Comme la formule d'intégration doit être vérifiée pour tout polynôme de degré 2n+1, et, en particulier, pour $Q_n(x).R_{n+1}(x)$ on doit avoir :

$$\int_{-1}^{+1} Q_n(x)\; R_{n+1}(x)\;dx = \sum_{i=0}^{n} A_i\, \phi(x_i) = 0$$

avec $\phi(x_i) = Q_n(x_i).R_{n+1}(x_i)$

il faut donc que :

$$A_0\, Q_n(x_0).R_{n+1}(x_0) + A_1.Q_n(x_1).R_{n+1}(x_1) + \ldots +$$

$$+A_n.Q_n(x_n).R_{n+1}(x_n) = 0$$

et pour qu'il en soit ainsi, il faut et il suffit que $R_{n+1}(x0) = 0$, $R_{n+1}(x1) = 0$ ... $R_{n+1}(xn) = 0$, c'est-à-dire que les abscisses x_i soient les racines du polynôme R_{n+1} (solutions de l'équation $R_{n+1}(x) = 0$).

On vient de montrer que, pour que la formule d'intégration de Gauss soit exacte pour P_{2n+1}, il faut que les x_i soient les racines du polynôme de Legendre de degré n+1.

Le polynôme R(x) peut être construit par récurrence en se rappelant que ces polynômes doivent vérifier la relation :

$$\int_{-1}^{+1} R_j(x).R_n(x)\ dx = 0 \quad \text{pour tout } j \leqslant n$$

le terme de plus haut degré aura comme coefficient 1, donc $R_0(x) = 1$. Le polynôme $R_1(x)$ est de la forme $R_1(x) = x + c_0\ R_0$.

on doit vérifier :

$$\int_{-1}^{+1} R_0(x).R_1(x)\ dx = 0$$

ou encore $\int_{-1}^{+1} 1.(x + c_0\ R_0)\ dx = 0$

en décomposant : $\int_{-1}^{+1} x\ dx + \int_{-1}^{+1} c_0\ R_0\ dx = 0$

d'où $$c_0 = \frac{\int_{-1}^{+1} x\ dx}{\int_{-1}^{+1} dx} = 0$$

donc $R_1(x) = x$

et ainsi de suite :

$R_{n+1}(x) = x^{n+1} + c_n\ R_n(x) + C_{n-1}\ R_{n-1}(x) \ldots + c_0\ R_0$

et toujours en vérifiant, pour tout $j \leqslant n$

$$\int_{-1}^{+1} R_{n+1}(x).R_j(x)\ dx = 0$$

c'est-à-dire $$\int_{-1}^{+1} x^{n+1}\ R_j(x)\ dx + c_j \int_{-1}^{+1} R_j^2(x)\ dx = 0$$

$(j \leqslant n)$

d'où $$a_j = \frac{-\int_{-1}^{+1} x^{n+1}\ R_j(x)\ dx}{\int_{-1}^{+1} R_j^2(x)\ dx}$$

ainsi : $R_0 = 1$

$R_1 = x$

$R_2 = x^2 - \frac{1}{6}$

$R_3 = x^3 - \frac{6}{10} x$

un autre procédé de construction donne (n+1) comme coefficient du terme de plus haut degré.

dans ces conditions $\sum_{i=0}^{n} a_i = 1$

Les polynômes de Legendre R_n possèdent d'autres propriétés, on montre que :

- les n racines sont distinctes et comprises entre -1 et +1
- ils vérifient la relation de récurrence :

  $R_0 = 1$, $R_1 = x$

  $(n+1)\ R_{n+1}(x) - (2n+1)x\ R_n(x) + n\ R_{n-1}(x) = 0$
- ils sont solutions de l'équation différentielle :

  $(x^2+1)y'' + 2xy' - n(n+1) = 0$
- l'intégrale :

  $$\int_{-1}^{+1} R_n^2(x)\ dx = \frac{2}{2n+1}$$

Il ne reste plus qu'à déterminer les coefficients A_i :

- à partir de la formule d'intégration :

  $$\int_{-1}^{+1} f(x)\ dx \simeq \sum_{i=0}^{n} A_i\ f(x_i)$$

et du polynôme d'interpolation de Lagrange de f(x) sur les points x_i :

$$f(x) \simeq P_n(x) = \sum_{i=0}^{n} f(x_i)\ \frac{L_i(x)}{L_i(x_i)}$$

avec $L_i(x) = (x-x_0)(x-x_1)\text{-}(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)$

après intégration :

$$\int_{-1}^{+1} f(x)\ dx \simeq \int_{1}^{+1} P_n(x)\ dx = \sum_{i=0}^{n} \frac{f(x_i)}{L_i(x_i)} \int_{-1}^{+1} L_i(x)\ dx$$

d'où, en rapprochant les relations :

$$\sum A_i f(x_i) = \sum \frac{f(x_i)}{Li(xi)} \int_{-1}^{+1} Li(x)\ dx$$

$$A_i = \frac{1}{L_i(x_i)} \int_{-1}^{+1} L_i(x)\ dx$$

Cette formule peut se mettre sous une autre forme en remarquant que $\frac{R_{n+1}(x)}{x-x_i}$ est un polynôme nul pour $x = x_j$ $(j \neq i)$ et pour $x = x_i$ est égal à la limite quand x tend vers x_i ;

or : $$\lim_{x \to x_i} \frac{R_{n+1}(x)}{x-x_i} = R'_{n+1}(x_i)$$

d'où : $$A_i = \frac{1}{R'_{n+1}(x_i)} \int_{-1}^{+1} \frac{R_{n+1}(x)}{x-x_i}\ dx$$

Il n'est pas nécessaire de refaire tous ces calculs à chaque fois que l'on calcule une intégrale définie par la méthode de Gauss, il suffit de consulter les tables de x_i et A_i et d'appliquer la formule :

$$\int_{-1}^{+1} f(x)\ dx \simeq \sum_{i=0}^{n} A_i\ f(x_i)$$

Lorsque l'intervalle est $[a,b]$ au lieu de $[-1,+1]$, il faut procéder au changement de variable suivant :

$$t = \frac{a+b}{2} + x\ \frac{b-a}{2} \qquad \begin{matrix} (t = a \text{ quand } x = -1 \\ t = b \text{ quand } x = +1) \end{matrix}$$

qui conduit à la formule modifiée.

$$\int_{a}^{b} f(t)\ dt \simeq \frac{b-a}{2} \sum_{i=0}^{i=n} A_i\ f\left(\frac{a+b}{2} + x\ \frac{b-a}{2}\right)$$

n = 2

$x_0=x_2=0.774\ 597$, $x_1=0$

$A_0=A_2=0.555\ 556$, $A_1=0.888\ 889$

n = 3

$-x_0=x_3=0.861\ 136$, $-x_1=x_2=0.339\ 981$

$A_0=A_3=0.347\ 855$, $A_1=A_2=0.652\ 145$

n = 4

$-x_0=x_4=0.906\ 180$, $-x_1=x_3=0.538\ 469$, $x_2=0$

$A_0=A_4=0.236\ 927$, $A_1=A_3=0.478\ 629$, $A_2=0.568\ 889$

n = 5

$-x_0=x_5=0.932470$, $-x_1=x_4=0.661\ 209$, $-x_2=x_3=0.238\ 619$

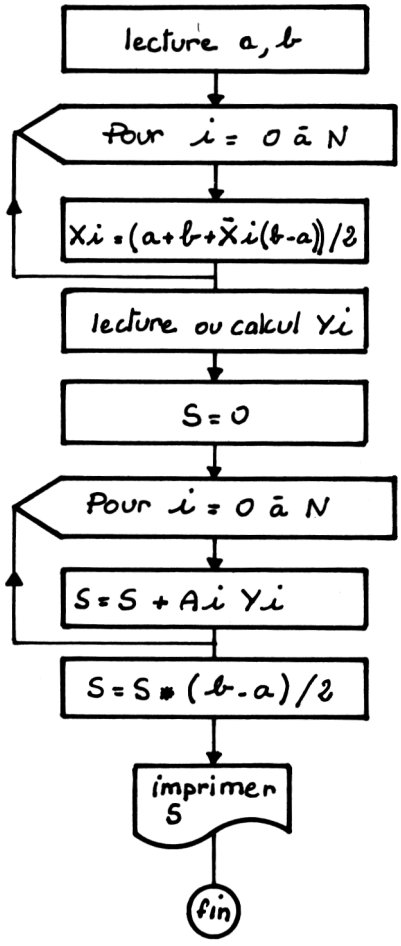
$A_0=A_5=0.171\ 324$, $A_1=A_4=0.360\ 762$, $A_2=A_3=0.467\ 914$

Exercice :

a - construire, au moyen de la formule de récurrence les polynômes R_n

b - rechercher les racines en utilisant une méthode décrite au chapitre II

c - calculer les coefficients A_i *en établissant une formule d'intégration*


lecture a, b
Pour i = 0 à N
Xi = (a+b+X̄i(b-a))/2
lecture ou calcul Yi
S = 0
Pour i = 0 à N
S = S + Ai Yi
S = S * (b-a)/2
imprimer S
fin

```
10 REM INTEGRATION  ** METHODE DE GAUSS **
15 NM=4
20 INPUT "INTERVALLE A,B";AA,BB:INPUT "NB DE POINTS";NP
21 LPRINT "INTERVALLE A,B";AA;BB,"NB DE POINTS";NP
25 N=NP-1
30 DIM X(N),Y(N),A(N),TX(NM,NM),TA(NM,NM)
35 LPRINT "Y=1/X"
40 IF N)NM PRINT "NB POINTS TROP GRAND MAXI="NM+1:END
50 FOR K=0 TO NM:FOR L=0 TO NM:READ TX(K,L):NEXT L,K
60 FOR K=0 TO NM:FOR L=0 TO NM:READ TA(K,L):NEXT L,K
70 REM CALCUL DES X(I)
80 FOR I=0 TO N:X(I)=(AA+BB+TX(N,I)*(BB-AA))/2:NEXT I
90 REM CALCUL DES Y(I)
100 FOR I=0 TO N:XX=X(I):GOSUB 1000:Y(I)=YY:NEXT I
110 CLS:FOR I=0 TO N:LPRINT"X=";X(I),"Y=";Y(I):NEXT I
120 REM CALCUL DES A(I)
130 FOR I=0 TO N:A(I)=TA(N,I):NEXT I
140 REM INTEGRATION
150 S=0:FOR I=0 TO N:S=S+A(I)*Y(I):NEXT I
160 S=S*(BB-AA)/2
165 LPRINT "INTEGRALE DE Y=1/X  (  LOG(X)  )"
170 LPRINT"J SUR";AA;",";BB;"=";S:END
1000 YY=1/XX:RETURN
3000 DATA 0,0,0,0,0,0,0,0,0,0
3001 DATA -0.774597,0,0.774597,0
3002 DATA 0,-0.861136,-0.339981,0.339981,0.861136
3003 DATA 0,-0.90618,-0.538469,0,0.538469,0.90618
3004 DATA 0,0,0,0,0,0,0,0,0,0
3005 DATA 0.555556,0.888889,0.555556,0
3006 DATA 0,0.347855,0.652145,0.652145,0.347855,0
3007 DATA 0.236927,0.478629,0.568889
3008 DATA 0.478629,0.236927
3009 DATA -0.906180,-0.538469,0.538469,0.90618
```

```
INTERVALLE A,B 1   4                NB DE POINTS 3
Y=1/X
X= 1.3381          Y= .747326
X= 2.5             Y= .4
X= 3.6619          Y= .273083
INTEGRALE DE Y=1/X   (  LOG(X)   )
J SUR 1 , 4 = 1.38367


INTERVALLE A,B 1   4                NB DE POINTS 4
Y=1/X
X= 1.2083          Y= .827612
X= 1.99003         Y= .502505
X= 3.00997         Y= .332229
X= 3.7917          Y= .263734
INTEGRALE DE Y=1/X   (  LOG(X)   )
J SUR 1 , 4 = 1.386

INTERVALLE A,B 1   4                NB DE POINTS 5
Y=1/X
X= 1.14073         Y= .876632
X= 1.6923          Y= .590913
X= 2.5             Y= .4
X= 3.3077          Y= .302325
X= 3.85927         Y= .259116
INTEGRALE DE Y=1/X   (  LOG(X)   )
J SUR 1 , 4 = 1.38626
```

METHODE DE ROMBERG

Cette méthode d'"accélération de la convergence" comprend deux étapes :

1 - calcul de $\int_a^b f(x)\,dx$ par la méthode des trapèzes en utilisant successivement des pas :

$h_0 = b-a$, $h_1 = \frac{h_0}{2}$, $h_2 = \frac{h_0}{2^2}$ ... $h_k = \frac{h_0}{2^k}$

2 - extrapolation répétée ("accélération de convergence") sur les évaluations (h_0, T_0^0), (h_1, T_0^1)...(h_k^2, T_0^k) où T_0^k est le résultat de l'intégration donné par la méthode des trapèzes pour le pas h_k

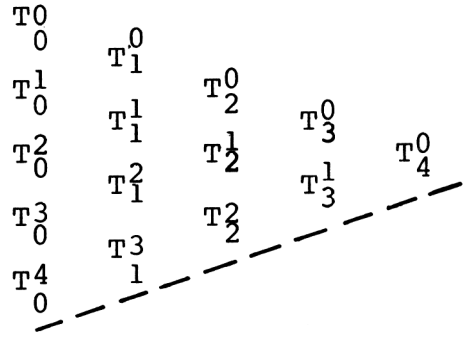
L'extrapolation est donnée par la formule récurrente :

$$T_0^k = T\left(\frac{b-a}{2^k}\right)$$

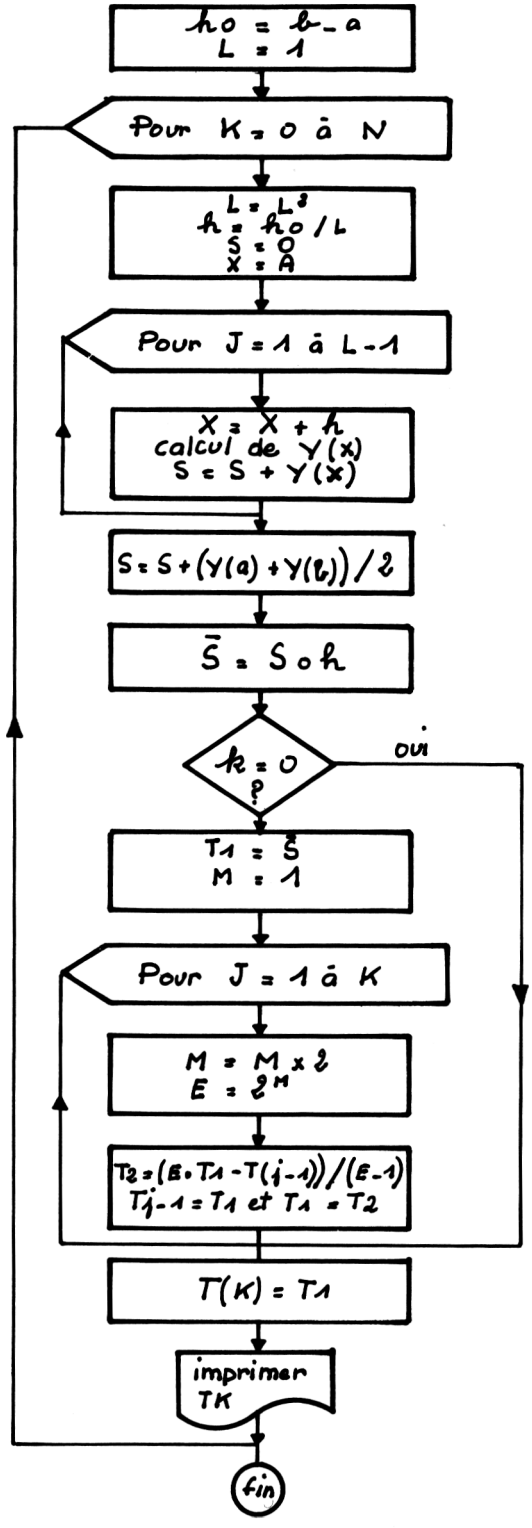
$$T_j^k = \frac{2^{2j}\ T_{j-1}^{k+1} - T_{j-1}^k}{2^{2j}-1}$$

On montre que le résultat donné par T_n^0 est exact pour les polynômes de degré 2n-1.

Les valeurs T_j^k peuvent être calculées en utilisant le tableau :



En examinant le calcul des coefficients de ce tableau, on voit qu'il n'est pas nécessaire de stocker tous les coefficients pour arriver à T_n^0. Le calcul peut être arrêté lorsque la précision est atteinte.


ho = b - a
L = 1
Pour K = 0 à N
L = L²
h = ho / L
S = 0
X = A
Pour J = 1 à L-1
X = X + h
calcul de Y(x)
S = S + Y(x)
S = S + (Y(a) + Y(b)) / 2
S̄ = S o h
k = 0 ?
oui
T1 = S̄
M = 1
Pour J = 1 à K
M = M x 2
E = 2^M
T2 = (E.T1 - T(j-1)) / (E-1)
Tj-1 = T1 et T1 = T2
T(K) = T1
imprimer TK
fin

```
5 REM  METHODE DE ROMBERG
10 DEFINT I-N
20 INPUT "INTERVALLE A,B";A,B:INPUT "NB CALCULS";N
25 LPRINT "INTERVALLE A,B ";A;B
26 LPRINT "NB DE CALCULS ";N
30 DIM T(N)
40 H0=B-A:L=1:X=A:GOSUB 1000:YA=Y:X=B:GOSUB 1000:YB=Y
45 LPRINT "K","H","S","T"
50 FOR K=0 TO N:L=L*2:H=H0/L:S=0:X=A
60 FOR J=1 TO L-1:X=X+H:GOSUB 1000:S=S+Y:NEXT J
70 S=S+(YA+YB)/2:ST=S*H
80 IF K=0 THEN T(0)=ST:GOTO 160
90 T1=ST:M=1:FOR J=1 TO K
100 M=M*2:E=2↑M:T2=(E*T1-T(J-1))/(E-1)
110 T(J-1)=T1:T1=T2:NEXT J:T(K)=T1
160 LPRINT K,H,ST,T(K)
170 NEXT K:END
1000 Y=1/X:RETURN
```

```
INTERVALLE A,B  1  4
NB DE CALCULS  5
K                 H                 S                 T
 0                 1.5               1.5375            1.5375
 1                 .75               1.42809           1.39162
 2                 .375              1.39713           1.38648
 3                 .1875             1.38903           1.3863
 4                 .09375            1.38698           1.38629
 5                 .046875           1.38647           1.38629
```

EQUATIONS DIFFERENTIELLES

PROBLEME DE CAUCHY

Une équation différentielle du premier ordre peut être mise sous forme résolue :

$$\frac{dy}{dx} = f(x,y) \qquad (\text{ou } y' = f(x,y))$$

où on recherche la fonction y à partir de sa dérivée.

Il existe une infinité de solutions qui dépendent d'une constante arbitraire. En fixant une valeur y_0 de la solution pour une valeur initiale x_0, on obtient une solution particulière.

Le problème de Cauchy (ou à conditions initiales) se résume comme suit :

soit x tel que $a \leqslant x \leqslant b$, trouver y(x) dans l'intervalle [a,b] vérifiant :

$$\begin{cases} y'(x) = f(x,y) \\ y(a) = \alpha \end{cases}$$

On démontre que ce problème admet une solution unique si f satisfait la condition de Lipschitz :

$$|f(x,y_1)-f(x,y_2)| \leqslant L\ |y_1-y_2|$$

où L est une constante positive.

On démontre aussi que si f'_y (dérivée de la fonction f par rapport à y) est continue et bornée, alors f satisfait la condition de Lipschitz.

D'une manière plus générale, on définit des systèmes du premier ordre.

$$\begin{aligned} y'_1 &= f_1(x, y_1, y_2 \ldots y_n) \\ y'_2 &= f_2(x, y_1, y_2 \ldots y_n) \\ &\text{----------} \\ y'_n &= f_n(x, y_1, y_2 \ldots y_n) \end{aligned}$$

on cherche la solution $y_1, y_2 \ldots y_n$ telle que :

$$\text{et} \quad \begin{cases} Y' = F(x,Y) \\ Y(a) = \alpha \end{cases}$$

$$\text{avec} \quad Y' = \begin{vmatrix} y'_1 \\ y'_2 \\ \vdots \\ y'_n \end{vmatrix} \qquad F = \begin{vmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{vmatrix} \qquad \text{et} \quad Y = \begin{vmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{vmatrix}$$

Si l'équation différentielle est d'ordre supérieur (n), elle dépend de n constantes arbitraires. Pour obtenir une solution particulière, il faut donc imposer n conditions.

Le problème est à valeurs initiales, si on impose pour x_0 donné, $y(x_0)$, $y'(x_0)$, $y''(x_0)$ ... Lorsque les solutions sont données pour plusieurs x, c'est un problème avec conditions aux limites et les calculs sont très différents et ne seront pas traités ici.

Une équation différentielle d'ordre n peut se ramener à un système de n équations différentielles d'ordre 1. Par exemple :

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = \alpha, \; y'(a) = \beta \end{cases}$$

On pose alors u=y et v=y' et l'équation se transforme en un système différentiel :

$$\begin{cases} u' = v \\ v' = g(x, u, v) \end{cases}$$

avec comme conditions initales :

$$u(a) = \alpha \quad \text{et} \quad v(a) = \beta$$

Les méthodes présentées sont des méthodes discrètes : l'intervalle [a,b] est partagé en n parties égales définissant le "pas" d'intégration :

$$h = \frac{b-a}{n}$$

METHODE D'EULER

Cette méthode n'est en pratique jamais utilisée ; mais, son étude est intéressante pour bien comprendre les "méthodes à un pas".

Soit l'équation différentielle :

$$y' = f(x,y)$$

et une solution y_n au point x_n. La méthode permet de calculer une solution approchée au point x_{n+1} tel que $x_{n+1}-x_n = h$ (h étant le pas d'intégration).

La formule de calcul (ou schéma) peut être établie à partir du développement de Taylor au voisinage de x_n :

$$y(x_{n+1}) = y(x_n)+hy'(x_n)+\frac{h^2}{2}y''(x_n)+\frac{h^3}{6}y'''(x_n)+ \ldots$$

en s'arrêtant aux deux premiers termes :

$$y(x_{n+1}) = y(x_n)+hy'(x_n)+h^2\,\varepsilon(h)$$

où $h^2\varepsilon(h)$ est l'erreur de troncature provenant du développement limité.

La formule d'intégration d'Euler est donc :

$$y(x_{n+1}) \simeq y(x_n) + h\, y'(x_n)$$

en notant :

$$y(x_{n+1}) = y_{n+1},\ y(x_n) = y_n$$

et en remarquant que :

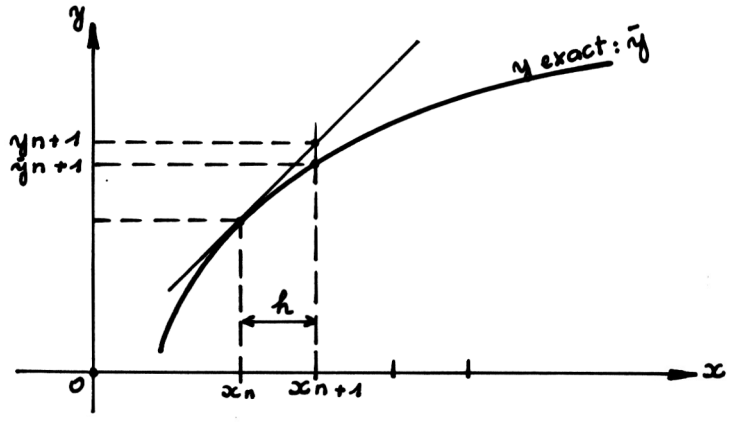
$$y'(x_n) = f(x_n, y_n) :$$

$$y_{n+1} \simeq y_n + h\, f(x_n, y_n)$$

Il est possible également de reprendre tout simplement l'expression approchée de la dérivée :

$$y' \simeq \frac{\Delta y}{\Delta x} = \frac{y_{n+1}-y_n}{h}$$

La représentation graphique de ce schéma montre bien que la courbe est remplacée localement par sa tangente :



l'erreur $|h^2\varepsilon| = |y_{n+1} - \bar{y}_{n+1}|$

L'exemple suivant montre le principe du calcul :

$$\begin{cases} y' = y \\ y(0) = 1 \end{cases}$$

la solution particulière, passant par le point (x = 0, y = 1) est $y = e^x$. L'application de la formule $y_{n+1} = y_n + h\, f_n$ donne :

x	y pour h=0.4	y pour h=0.2	y pour h=0.1	y pour h=0.05	y exact
1		2.4883	2.59374	2.6533	2.71828
2	5.378	6.19174	6.7275	7.03999	7.38905
3		15.407	17.4494	18.6792	20.0555
4	28.955	38.3376	45.2593	49.5614	54.598

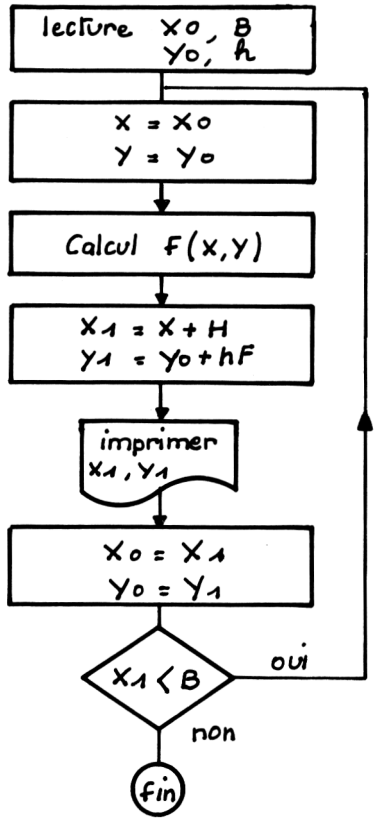
Exercice : *calculer pour x = 2 et pour x = 4 l'erreur* $\varepsilon = |y - y_{exact}|$ *pour chaque pas (h=0.4, h=0.2 ...) et représenter la courbe* ε *en fonction de h.*

L'erreur constatée est due non seulement à l'approximation de la courbe par sa tangente sur l'intervalle d'amplitude égale au pas, mais aussi par la limitation de chiffres significatifs pris en compte à chaque opération élémentaire.

On recherche donc des méthodes où l'erreur diminue quand le pas, lui aussi, diminue. Théoriquement, l'erreur doit s'annuler lorsque le pas est infiniment petit : les méthodes doivent être convergentes.

On exige en outre que les erreurs (sur les valeurs initiales et les imprécisions de calcul) ne s'amplifient pas au cours des calculs : les méthodes doivent être stables.

```
5 REM METHODE D'EULER
10 INPUT "INTERVALLE X0,B";X0,B
15 LPRINT "INTERVALLE X0,B ";X0;B
20 INPUT "VALEUR INITIALE Y0";Y0
25 LPRINT "VALEUR INITIALE Y0= ";Y0
30 INPUT "PAS H";H
31 LPRINT "PAS H= ";H
32 LPRINT "EQUATION DIFFERENTIELLE  Y'=-2*X*Y↑2"
35 CLS:LPRINT "X","YN","YEXACT"
40 X=X0:Y=Y0:GOSUB 1000
50 X1=X+H:X=X1:GOSUB 2000
60 Y1=Y0+H*F
70 LPRINT X1,Y1,YY
80 X0=X1:Y0=Y1
90 IF X1<B THEN GOTO 40
100 END
110 REM EQUATION DIFFERENTIELLE
1000 F=-2*X*Y*Y:RETURN
1500 REM SOLUTION EXACTE POUR COMPARAISON
2000 YY=1/(X*X+1):RETURN
```



Exemple : $y' = -2xy^2, y(0) = 1$

```
INTERVALLE X0,B  0  1
VALEUR INITIALE Y0=  1
PAS H=  .1
EQUATION DIFFERENTIELLE  Y'=-2*X*Y↑2
X                    YN                   YEXACT
 .1                   1                    .990099
 .2                   .98                  .961539
 .3                   .941584              .917431
 .4                   .888389              .862069
 .5                   .82525               .8
 .6                   .757147              .735294
 .7                   .688354              .671141
 .8                   .622018              .609756
 .9                   .560113              .552486
 1                    .503642              .5
```

ETUDE SOMMAIRE DES METHODES A UN PAS

Il n'est pas possible de donner ici la théorie complète de ces méthodes numériques. Néanmoins, pour mieux comprendre le principe de ces méthodes et avoir une idée des problèmes qui peuvent se présenter, les notions de schéma général, convergence, stabilité, consistance, ordre ... seront résumées sans démonstration.

SCHEMA GENERAL

Les méthodes à un pas utilisent la formule générale :

$$y_{n+1} = y_n + h\,\Phi(x_n, y_n, h)$$

Pour la méthode d'Euler, la fonction Φ est $f(x,y)$:

$$y_{n+1} = y_n + h\,f(x,y)$$

CONVERGENCE

La méthode converge sur l'intervalle $[a,b]$ si, pour la suite des y_n calculés, l'écart maximum avec la solution exacte diminue quand le pas h diminue :

$$\max_n |y_n - y(x)| \to 0 \text{ quand } h \to 0$$

pour $0 \leqslant n \leqslant N$, N étant le calcul effectué pour $x = b$.

STABILITE

La méthode est stable si une perturbation sur y_0 ou Φ n'entraîne qu'une petite perturbation sur la solution approchée, et cela quel que soit le pas h.

Souvent, lorsque la méthode n'est pas stable, l'amplification des erreurs est exponentielle et au bout de quelques calculs, les résultats entraînent des dépassements de capacité.

On montre que si Φ vérifie la condition de Lipschitz :

$$|\phi(x,y,h) - \phi(x,\bar{y},h)| \leqslant M\,|y-\bar{y}|$$

alors la méthode est stable.

CONSISTANCE

Le schéma de calcul $y_{n+1} = y_n + h\,\phi(x_n, y_n, h)$ est consistant avec l'équation différentielle si :

$$\max_n \left| \frac{1}{h} (y(x_{n+1}) - y(x_n)) - \phi(x_n, y(x_n), h) \right| \to 0$$

quand $h \to 0$

Cela signifie que le schéma doit être une approximation vraisemblable, bien construite.

On montre que le schéma est consistant si $\phi(x,y,0) = f(x,y)$.

Si la méthode est stable et consistante, alors elle converge pour n'importe quelle valeur initiale.

ORDRE

La méthode itérative est d'ordre p si pour toute solution :

$$\max_n \left| \frac{1}{h} (y(x_{n+1}) - y(x_n)) - \phi(x_n, y(x_n), h) \right| \leqslant Kh^p$$

Pour mieux comprendre ces notions théoriques, il est utile d'examiner, en détail, la méthode d'Euler :

$$y_{n+1} = y_n + h\, f(x_n, y_n)$$

La méthode est consistante si $\phi(x,y,0) = f(x,y)$ ici, c'est immédiat puisque $\phi = f(x,y)$ ne dépend pas du pas h : le schéma d'Euler est donc consistant.

La méthode est stable si :

$$|\phi(x,y,h) - \phi(x,\bar{y},h)| \leqslant M|y-\bar{y}|$$

immédiatement, on voit que la méthode d'Euler est stable, en effet $\phi(x,y,h) = f(x,y)$ or la fonction f, par hypothèse, satisfait la condition de Lipschitz : le schéma d'Euler est stable.

En conséquence, la méthode d'Euler est convergente.

Le schéma d'Euler a été établi à partir du développement de Taylor, limité aux termes en h :

$$\begin{aligned} y(x_{n+1}) &= y(x_n) + h\, y'(x_n) + h^2 \varepsilon(h) \\ &= y(x_n) + h\, f(x,y) + h^2 \varepsilon(h) \end{aligned}$$

d'où :

$$\begin{aligned} y(x_{n+1}) - y(x_n) - h\, f(x,y) &= h^2 \varepsilon(h) \\ \frac{1}{h} (y(x_{n+1}) - y(x_n)) - f(x,y) &= h\varepsilon(h) \end{aligned}$$

donc :

$$\left|\frac{1}{h}\,(y(x_{n+1})-y(x_n))-\phi(x_n,y_n,h)\right| \leqslant Kh$$

Le schéma d'Euler est d'ordre 1 (l'erreur est divisée par λ quand le pas est divisé par λ).

Pour l'exemple donné au paragraphe précédent, en traçant la courbe représentative de l'erreur ε = $|y_n - y \text{ exact}|$ en fonction du pas h, on constate que $\varepsilon(h)$ est presque une droite.

METHODES DE RUNGE KUTTA

Ces méthodes utilisent des schémas numériques de la forme :

$$k_1 = h\, f(x_n, y_n)$$
$$k_2 = h\, f(x_n+ah, y_n+\alpha k_1)$$
$$k_3 = h\, f(x_n+bh, y_n+\beta k_2)$$
$$\text{----------}$$
$$\text{et } y_{n+1} = y_n+R_1k_1+R_2k_2+ \ldots$$

Pour des erreurs nettement plus faibles que celles engendrées par la méthode d'Euler, les méthodes de Runge-Kutta ne sont pas beaucoup plus compliquées et la programmation est très simple.

FORMULES A DEUX APPROXIMATIONS

Elles sont de la forme :

$$k_1 = h\, f(x_n, y_n)$$
$$k_2 = h\, f(x_n+ah, y_n+\alpha k_1)$$
$$\text{et } y_{n+1} = y_n+R_1k_1+R_2k_2$$

Il nous faut déterminer les coefficients a, α, R_1, R_2. A partir de $y'=f(x,y)$, il est possible de calculer y^{II}, y^{III}, y^{IV}... si la fonction $f(x,y)$ est suffisamment régulière, sans oublier que f est une fonction composée :

$$f = f(x,y(x))$$

donc :

$$y'' = f'_x+y'_x.f'_y$$

En utilisant la formule de Taylor :

$$y(x_{n+1})-y(x_n) = h\, y'(x_n,y_n)+ \frac{h^2}{2} y''+\ldots$$

$$= h\, f(x_n,y_n)+ \frac{h^2}{2} (f'_x+y'f'_y)+\ldots$$

et la formule approchée :

$$y_{n+1}-y_n = R_1 h\, f(x_n,y_n)+R_2 h\, f(x_n+ah, y_n+\alpha k_1)$$

$$= R_1 h\, f(x_n,y_n)+R_2 h\, f(x_n,y_n)+R^2h^2\left[af'_x + \alpha \frac{k1}{h} f'_y\right]+.$$

l'identification des termes en h et h^2 donne :

$$R_1+R_2 = 1$$

$$aR_1 = \frac{1}{2}$$

$$\alpha R_2 = 1$$

d'où $\alpha = a$, $R_1 = \frac{2a-1}{2a}$, $R_2 = \frac{1}{2a}$

a est arbitraire (il y a plus d'inconnues que d'équations).

. si a = 1/2, $R_1 = 0$, $R_2 = 1$, c'est la formule de la tangente améliorée :

$$y_{n+1/2} = y_n+ \frac{h}{2} f(x_n,y_n)$$

$$y_{n+1} = y_n+h\, f(x_n+1/2h,\ y_n+1/2)$$

. si a = 1, $R_1 = 1/2$, $R_2 = 1/2$, c'est la formule d'Euler Cauchy :

$$y_{n1} = h\, f(x_n,y_n)$$

$$y_{n+1} = y_n + \frac{h}{2} (f(x_n,y_n)+f(x_{n+1},y_{n1}))$$

Ces schémas sont d'ordre 2.

En appliquant le même procédé, on arrive à la méthode classique d'ordre 4, dont le schéma de calcul est le suivant :

$$k_1 = h\, f(x_n,y_n)$$

$$k_2 = h\, f(x_n+h/2,\ y_n+k1/2)$$

$$k_3 = h\, f(x_n+h/2,\ y_n+k2/2)$$

$$k_4 = h\, f(x_n+h,\ y_n+k_3)$$

et $$y_{n+1} = y_n+ \frac{1}{6} (k_1+2k_2+2k_3+k_4)$$

```
5 REM METHODE EULER AMELIOREE
10 INPUT "INTERVALLE X0,B";X0,B
15 LPRINT "INTERVALLE X0,B ";X0;B
20 INPUT "VALEUR INITIALE Y0";Y0
25 LPRINT "VALEUR INITIALE Y0=";Y0
30 INPUT "PAS H";H
31 LPRINT "PAS H= ";H
32 LPRINT "EQUATION DIFFERENTIELLE Y'=-2*X*Y↑2"
35 CLS:LPRINT "X","YN","YEXACT"
40 X=X0:Y=Y0:GOSUB 1000
50 X1=X+H:X=X1:GOSUB 2000
60 YP=Y0+H*F/2
70 X=X0+H/2:Y=YP:GOSUB 1000
80 Y1=Y0+H*F
90 LPRINT X1,Y1,YY
100 X0=X1:Y0=Y1
110 IF X1<B THEN GOTO 40
120 END
130 REM EQUATION DIFFERENTIELLE
1000 F=-2*X*Y*Y:RETURN
1500 REM SOLUTION EXACTE POUR COMPARAISON
2000 YY=1/(X*X+1):RETURN
```

```
INTERVALLE X0,B  0  1
VALEUR INITIALE Y0= 1
PAS H=  .1
EQUATION DIFFERENTIELLE Y'=-2*X*Y↑2
X                YN               YEXACT
 .1               .99              .990099
 .2               .961176          .961539
 .3               .916742          .917431
 .4               .861104          .862069
 .5               .798888          .8
 .6               .73418           .735294
 .7               .670145          .671141
 .8               .608952          .609756
 .9               .551905          .552486
 1                .499638          .5
```

```
5 REM METHODE DE RUNGE-KUTTA
10 INPUT "INTERVALLE X0,B";X0,B
15 LPRINT "INTERVALLE X0,B ";X0;B
20 INPUT "VALEUR INITIALE Y0";Y0
25 LPRINT "VALEUR INITIALE Y0= ";Y0
30 INPUT "PAS H";H
31 LPRINT "PAS H=";H
32 LPRINT "EQUATION DIFFERENTIELLE  Y'=-2*X*Y↑2"
35 CLS:LPRINT "X","YN","YEXACT"
40 X=X0:Y=Y0:GOSUB 1000
50 K1=H*F
60 X=X0+H/2:Y=Y0+K1/2:GOSUB 1000
70 K2=H*F
80 Y=Y0+K2/2:GOSUB 1000
90 K3=H*F
100 X=X0+H:Y=Y0+K3:GOSUB 1000:GOSUB 2000
110 K4=H*F
120 Y1=Y0+(K1+2*K2+2*K3+K4)/6
125 X1=X0+H
130 LPRINT X1,Y1,YY
140 X0=X1:Y0=Y1
150 IF X1<B THEN GOTO 40
160 END
170 REM EQUATION DIFFERENTIELLE
1000 F=-2*X*Y*Y:RETURN
1500 REM SOLUTION EXACTE POUR COMPARAISON
2000 YY=1/(X*X+1):RETURN
```

```
INTERVALLE X0,B  0  1
VALEUR INITIALE Y0=  1
PAS H= .1
EQUATION DIFFERENTIELLE  Y'=-2*X*Y↑2
X                 YN                YEXACT
 .1                .990099           .990099
 .2                .961538           .961539
 .3                .917431           .917431
 .4                .862068           .862069
 .5                .799999           .8
 .6                .735294           .735294
 .7                .671141           .671141
 .8                .609756           .609756
 .9                .552487           .552486
 1                 .500001           .5
```

SYSTEMES DIFFERENTIELS

La formule de Runge Kutta se généralise et permet de résoudre aussi facilement des systèmes d'équations différentielles (ou équations différentielles d'ordre égal ou supérieur à 2).

En utilisant les notations vectorielles :

$$K1 = h\ F\ (x_n, Y_n)$$

$$K2 = h\ F\ (x_n+h/2, Y_n+K1/2)$$

$$K3 = h\ F\ (x_n+h/2, Y_n+K2/2)$$

$$K4 = h\ F\ (x_n+h, Y_n+K3)$$

$$\text{et}\quad Y_{n+1} = Y_n + \frac{1}{6}\left[K1+2K2+2K3+K4\right]$$

avec (pour un système d'ordre 4)

$$K1 = (k_{11}, k_{12}, k_{13}, k_{14})$$

$$K2 = (k_{21}, k_{22}, k_{23}, k_{24})$$

$$K3 = (k_{31}, k_{32}, k_{33}, k_{34})$$

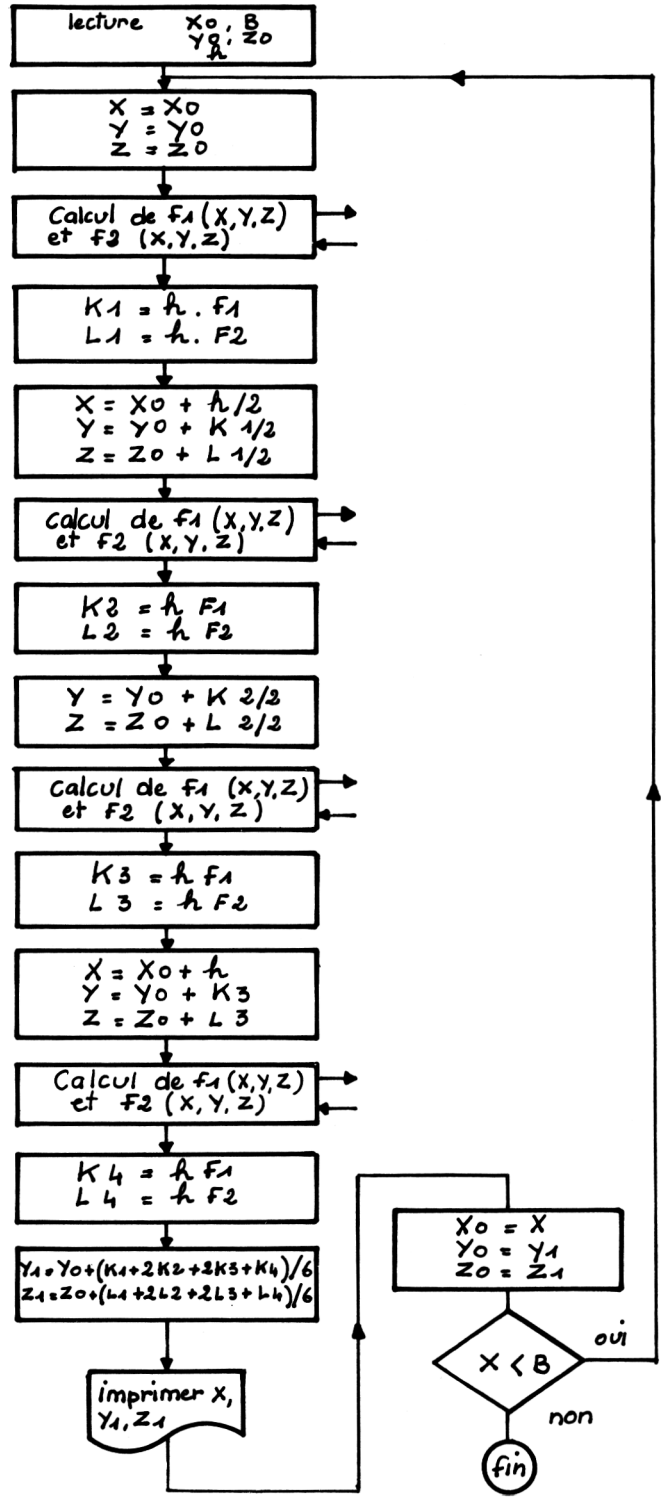
$$K4 = (k_{41}, k_{42}, k_{43}, k_{44})$$

$$Y_n = (y_{1n}, y_{2n}, y_{3n}, y_{4n})$$

$$F(x,Y) = (f_1(x,Y), f_2(x,Y), f_3(x,Y), f_4(x,Y))$$

L'exemple ci-après traite le cas de l'équation différentielle du deuxième ordre : $y''-4y = \frac{x^2}{10}$ avec $y(0)=0$ $y'(0)=0,1$ dont la solution particulière est :

$$y = (5e^{2x}-3e^{-2x}-4x^2-2)/160$$


lecture X0 ; B
Y0 ; Z0
h
X = X0
Y = Y0
Z = Z0
Calcul de F1 (X,Y,Z)
et F2 (X,Y,Z)
K1 = h . F1
L1 = h . F2
X = X0 + h/2
Y = Y0 + K 1/2
Z = Z0 + L 1/2
calcul de F1 (X,Y,Z)
et F2 (X,Y,Z)
K2 = h F1
L2 = h F2
Y = Y0 + K 2/2
Z = Z0 + L 2/2
calcul de F1 (X,Y,Z)
et F2 (X,Y,Z)
K3 = h F1
L3 = h F2
X = X0 + h
Y = Y0 + K3
Z = Z0 + L3
Calcul de F1 (X,Y,Z)
et F2 (X,Y,Z)
K4 = h F1
L4 = h F2
Y1 = Y0 + (K1 + 2K2 + 2K3 + K4)/6
Z1 = Z0 + (L1 + 2L2 + 2L3 + L4)/6
imprimer X,
Y1, Z1
X0 = X
Y0 = Y1
Z0 = Z1
X < B
oui
non
fin

```
5 REM SYSTEME DIFFERENTIEL  METHODE RUNGE-KUTTA
10 INPUT "INTERVALLE X0,B";X0,B
15 LPRINT "INTERVALLE X0,B ";X0;B
20 INPUT "VALEURS INITIALES Y0,Z0";Y0,Z0
25 LPRINT "VALEURS INITIALES Y0,Z0 ";Y0;Z0
30 INPUT "PAS H";H
31 LPRINT "PAS H= ";H
32 LPRINT "SYSTEME DIFFERENTIEL (Y''=4*Y+0.1*X↑2)"
33 LPRINT "    Y'=Z"
34 LPRINT "    Z'=4*Y+0.1*X↑2"
35 CLS:LPRINT "X","YN","ZN","YEXACT"
40 X=X0:Y=Y0:Z=Z0:GOSUB 1000
50 K1=H*F1:L1=H*F2
60 X=X0+H/2:Y=Y0+K1/2:Z=Z0+L1/2:GOSUB 1000
70 K2=H*F1:L2=H*F2
80 Y=Y0+K2/2:Z=Z0+L2/2:GOSUB 1000
90 K3=H*F1:L3=H*F2
100 X=X0+H:Y=Y0+K3:Z=Z0+L3:GOSUB 1000:GOSUB 2000
105 K4=H*F1:L4=H*F2
110 Y1=Y0+(K1+2*K2+2*K3+K4)/6
120 Z1=Z0+(L1+2*L2+2*L3+L4)/6
130 LPRINT X,Y1,Z1,YY
140 X0=X:Y0=Y1:Z0=Z1
150 IF X<B THEN GOTO 40
160 END
170 REM SYSTEME DIFFERENTIEL
1000 F1=Z:F2=4*Y+X*X/10:RETURN
1500 REM SOLUTION EXACTE POUR COMPARAISON
2000 YY=(5*EXP(2*X)-3*EXP(-2*X)-4*X*X-2)/160:RETURN
```

```
INTERVALLE X0,B  0  1
VALEURS INITIALES Y0,Z0  0  .1
PAS H=  .1
SYSTEME DIFFERENTIEL (Y''=4*Y+0.1*X↑2)
   Y'=Z
   Z'=4*Y+0.1*X↑2
```

X	YN	ZN	YEXACT
.1	.0100675	.10204	.0100676
.2	.0205507	.108376	.020551
.3	.0319005	.119463	.031901
.4	.0446226	.135945	.0446232
.5	.0592976	.158687	.0592986
.6	.0766049	.1888	.0766063
.7	.0973494	.227694	.0973513
.8	.122494	.277131	.122497
.9	.153199	.339295	.153202
1	.190866	.416882	.190871

```
INTERVALLE X0,B  0  1
VALEURS INITIALES Y0,Z0  0  .1
PAS H=  .05
SYSTEME DIFFERENTIEL (Y''=4*Y+0.1*X↑2)
   Y'=Z
   Z'=4*Y+0.1*X↑2
```

X	YN	ZN	YEXACT
.05	5.00839E-03	.100505	5.00839E-03
.1	.0100676	.10204	.0100676
.15	.0152302	.104647	.0152302
.2	.020551	.108376	.020551
.25	.0260876	.11329	.0260876
.3	.031901	.119463	.031901
.35	.0380563	.126981	.0380563
.4	.0446232	.135946	.0446232
.45	.0516769	.146472	.0516769
.5	.0592985	.158688	.0592986
.55	.0675763	.172743	.0675764
.6	.0766062	.188802	.0766063
.65	.086493	.207051	.0864931
.7	.0973512	.227697	.0973514
.75	.109306	.250973	.109307
.8	.122497	.277135	.122497
.85	.137073	.306472	.137073
.9	.153202	.339301	.153202
.95	.171067	.375977	.171067
1	.19087	.416891	.190871

APPLICATIONS

Deux problèmes sont traités ci-après pour montrer l'intérêt des programmes de calcul numérique qui viennent d'être décrits.

RESEAUX ELECTRIQUES EN COURANT CONTINU

Ces circuits ne comprennent que des résistances et des générateurs.

Une résistance R est caractérisée par sa valeur en ohm (Ω) et un générateur E, par sa force électromotrice en volt (V).

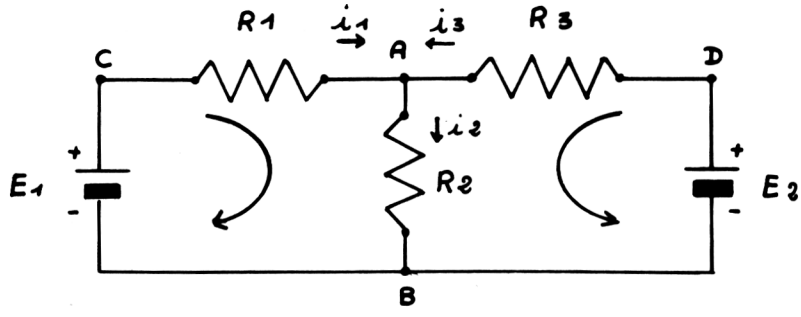
Un réseau de résistances et générateurs est constitué de mailles (contour fermé).

Les lois de Kirchhoff permettent de calculer les tensions aux noeuds et courants passant dans chaque branche.

1ère loi : la somme algébrique des courants qui parviennent à chaque noeud du réseau est nulle.

2ème loi : le long d'un contour fermé, orienté arbitrairement, la somme algébrique des tensions est nulle.

Exemple :



- loi des noeuds appliquée au point A(1ère loi) :

$$i_1+i_3-i_2 = 0$$

- loi des mailles appliquée à la maille ABCA

$$i_1R_1+i_2R_2-E_1 = 0$$

- loi des mailles appliquée à la maille ABDA

$$i_2R_2-E_2+i_3R_3 = 0$$

ces équations correspondent au système linéaire :

$$\begin{array}{llll} i_1 & -i_2 & +i_3 & = 0 \\ R_1i_1 & +R_2i_2 & +0 & = E_1 \\ 0 & +R_2i_2 & +R_3i_3 & = E_2 \end{array}$$

ou encore sous forme matricielle :

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & R_2 & 0 \\ 0 & R_2 & R_3 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} = \begin{bmatrix} 0 \\ E_1 \\ E_2 \end{bmatrix}$$

application numérique : $R_1 = 2\,\Omega$, $R_2 = 4\,\Omega$, $R_3 = 6\,\Omega$,
$E_1 = 12V$, $E_2 = 10V$

Exercice : *utiliser le programme de résolution des équations linéaires pour déterminer i_1, i_2, i_3 (on trouve $i_1 = 1.81818$, $i_2 = 2.09091$, $i_3 = 0.27273$).*

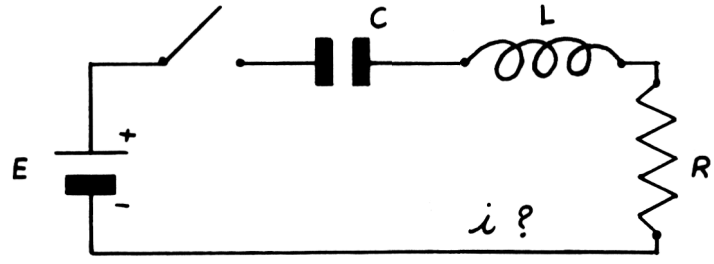
Cette méthode peut évidemment s'étendre à des circuits beaucoup plus complexes comportant un grand nombre de noeuds. Pour éviter la mise en équation fastidieuse, il existe des programmes de traitement des graphes.

CHARGE D'UN CONDENSATEUR

L'exemple suivant fait appel à la méthode de Runge Kutta pour les calculs de solutions du problème de Cauchy.

Il s'agit ici de l'étude classique du régime transitoire d'un circuit RLC ; les équations peuvent se résoudre assez facilement dans le cas d'une excitation échelon ou périodique, la méthode numérique permet de traiter facilement tous les cas.

Le circuit comprend une résistance R, une inductance L et un condensateur C en série. On veut calculer l'évolution du courant i lorsqu'on ferme l'interrupteur :



La loi de Kirchhoff appliquée à la maille donne :

$$E - \frac{q}{c} - L\frac{di}{dt} - R_i = 0$$

$$\text{et } i = \frac{dq}{dt}$$

c'est un système différentiel qui peut être mis sous la forme :

$$q' = i$$

$$i' = (E - q/c - R_i)/L$$

Exercice : *utiliser le programme de Runge Kutta pour les systèmes différentiels (chapitre VII) avec :*

(a)- E=10V, C=1μF, L=0.25H, R=100Ω

(b)- E=10V, C=1μF, L=0.25H, R=1000Ω

(c)- E=10V, C=1μF, L=0.25H, R=2000Ω

Ce problème est classique et le régime est différent suivant les racines de l'équation caractéristique de l'équation différentielle :

(a)- si $R^2-4\frac{L}{C}<0$, régime oscillatoire amorti

(b)- si $R^2-4\frac{L}{C}=0$, régime amorti critique

(c)- si $R^2-4\frac{L}{C}>0$, régime apériodique.

```
5 REM CHARGE D'UN CONDENSATEUR
16 DIM VV(127)
25 Y0=0:Z0=0
26 INPUT "E,R,L,C";E,R,L,C
27 LPRINT "TENSION E";E;"V"
28 LPRINT "RESISTANCE R";R;"OHM"
29 LPRINT "SELF L";L;"H"
30 LPRINT "CAPACITE C";C;"F"
31 INPUT "PAS H";H
32 LPRINT "PAS H=";H
35 LPRINT "T","I"
36 X0=0:B=TT
38 FOR K=0 TO 127
40 X=X0:Y=Y0:Z=Z0:GOSUB 1000
50 K1=H*F1:L1=H*F2
60 X=X0+H/2:Y=Y0+K1/2:Z=Z0+L1/2:GOSUB 1000
70 K2=H*F1:L2=H*F2
80 Y=Y0+K2/2:Z=Z0+L2/2:GOSUB 1000
90 K3=H*F1:L3=H*F2
100 X=X0+H:Y=Y0+K3:Z=Z0+L3:GOSUB 1000
105 K4=H*F1:L4=H*F2
110 Y1=Y0+(K1+2*K2+2*K3+K4)/6
120 Z1=Z0+(L1+2*L2+2*L3+L4)/6
130 LPRINT X,Z1
135 VV(K)=Z1
140 X0=X:Y0=Y1:Z0=Z1
150 NEXT K
160 END
170 REM SYSTEME DIFFERENTIEL
1000 F1=Z:F2=(E-Y/C-R*Z)/L:RETURN
```

```
TENSION E 10 V
RESISTANCE R 100 OHM
SELF L .25 H
CAPACITE C 1E-06 F
PAS H= 2E-04
T                    I
 2E-04                7.48356E-03
 4E-04                .013257
 6E-04                .0165761
 8E-04                .017126
 1E-03                .0150363
 1.2E-03              .0108268
 1.4E-03              5.29868E-03
 1.6E-03             -6.08153E-04
 1.8E-03             -5.96877E-03
 2E-03               -.0100121
 2.2E-03             -.0122263
 2.4E-03             -.0124159
 2.6E-03             -.010708
 2.8E-03             -7.50724E-03
 3E-03               -3.41393E-03
 3.2E-03              8.82556E-04
 3.4E-03              4.71498E-03
 3.6E-03              7.53775E-03
 3.8E-03              9.00037E-03
 4E-03                8.98554E-03
 4.2E-03              7.60906E-03
 4.4E-03              5.18435E-03
 4.6E-03              2.15971E-03
 4.8E-03             -9.60023E-04
 5E-03               -3.69438E-03
 5.2E-03             -5.65828E-03
 5.4E-03             -6.61308E-03
 5.6E-03             -6.49153E-03
 5.8E-03             -5.39478E-03
 6E-03               -3.56413E-03
 6.2E-03             -1.33362E-03
 6.4E-03              9.2772E-04
 6.6E-03              2.87456E-03
 6.8E-03              4.2358E-03
 7E-03                4.85001E-03
 7.2E-03              4.68143E-03
 7.4E-03              3.8158E-03
 7.6E-03              2.43798E-03
 7.8E-03              7.96295E-04
```

```
TENSION E 10 V
RESISTANCE R 1000 OHM
SELF L .25 H
CAPACITE C 1E-06 F
PAS H= 2E-04
T                    I
 2E-04                5.35467E-03
 4E-04                7.17954E-03
 6E-04                7.21974E-03
 8E-04                6.45349E-03
 1E-03                5.40802E-03
 1.2E-03              4.35065E-03
 1.4E-03              3.40278E-03
 1.6E-03              2.60712E-03
 1.8E-03              1.96629E-03
 2E-03                1.46467E-03
 2.2E-03              1.0801E-03
 2.4E-03              7.89928E-04
 2.6E-03              5.73699E-04
 2.8E-03              4.14193E-04
 3E-03                2.97509E-04
 3.2E-03              2.12747E-04
 3.4E-03              1.5154E-04
 3.6E-03              1.07568E-04
 3.8E-03              7.61202E-05
 4E-03                5.3717E-05
 4.2E-03              3.78127E-05
 4.4E-03              2.6557E-05
 4.6E-03              1.86135E-05
 4.8E-03              1.30208E-05
 5E-03                9.09275E-06
 5.2E-03              6.33927E-06
 5.4E-03              4.41284E-06
 5.6E-03              3.06806E-06
 5.8E-03              2.13016E-06
 6E-03                1.47744E-06
 6.2E-03              1.02351E-06
 6.4E-03              7.08172E-07
 6.6E-03              4.89749E-07
 6.8E-03              3.38549E-07
 7E-03                2.33789E-07
 7.2E-03              1.61405E-07
 7.4E-03              1.1119E-07
 7.6E-03              7.6451E-08
 7.8E-03              5.27514E-08
```

TENSION E 10 V
RESISTANCE R 2000 OHM
SELF L .25 H
CAPACITE C 1E-06 F
PAS H= 2E-04

T	I
2E-04	3.60533E-03
4E-04	4.22641E-03
6E-04	4.06734E-03
8E-04	3.72804E-03
1E-03	3.36943E-03
1.2E-03	3.03253E-03
1.4E-03	2.72584E-03
1.6E-03	2.44922E-03
1.8E-03	2.2004E-03
2E-03	1.97679E-03
2.2E-03	1.77589E-03
2.4E-03	1.5954E-03
2.6E-03	1.43325E-03
2.8E-03	1.28758E-03
3E-03	1.15672E-03
3.2E-03	1.03915E-03
3.4E-03	9.33538E-04
3.6E-03	8.38657E-04
3.8E-03	7.5342E-04
4E-03	6.76846E-04
4.2E-03	6.08054E-04
4.4E-03	5.46254E-04
4.6E-03	4.90735E-04
4.8E-03	4.40859E-04
5E-03	3.96052E-04
5.2E-03	3.55799E-04
5.4E-03	3.19637E-04
5.6E-03	2.87151E-04
5.8E-03	2.57966E-04
6E-03	2.31747E-04
6.2E-03	2.08193E-04
6.4E-03	1.87033E-04
6.6E-03	1.68024E-04
6.8E-03	1.50947E-04
7E-03	1.35605E-04
7.2E-03	1.21822E-04
7.4E-03	1.09441E-04
7.6E-03	9.83178E-05
7.8E-03	8.83255E-05

BIBLIOGRAPHIE SOMMAIRE

A. Ralston et H. S. Wilf :
Méthodes mathématiques pour calculateurs arithmétiques
Dunod 1965

G. Hacques :
Mathématiques pour l'informatique 3. Algorithmique numérique - (1) - Cours -
A. Colin 1971

P. Baudoux - O. Beaufays :
Théorie des circuits
Dunod 1968

Achevé d'imprimer en mars 1982
sur les presses de l'imprimerie Laballery et Cie
58500 Clamecy
Dépôt légal : mars 1982

N° d'impression : 20433
N° d'édition : 86595-29-2
ISBN : 2-86595-029-8

GUIDE PRATIQUE

* *

MÉTHODES DE CALCUL NUMÉRIQUE

PROGRAMMES EN BASIC

Cet ouvrage présente les méthodes classiques de calcul numérique. Il s'agit d'un guide pratique où, pour un même problème, les différentes méthodes utilisables sont présentées. Chaque algorithme est illustré par un organigramme, un programme en Basic et un exemple d'exécution. Deux exemples d'utilisation des méthodes proposées dans l'étude des circuits électriques sont développés.

TOME 1

Graphisme : Françoise Guillot

Editions du P.S.I.
Boîte Postale 86
F - 77400 Lagny/Marne

75.00

ISBN 2-86595-029-8 — Imprimé en France

GUIDE PRATIQUE

CLAUDE NOWAKOWSKI

METHODES DE CALCUL NUMERIQUE